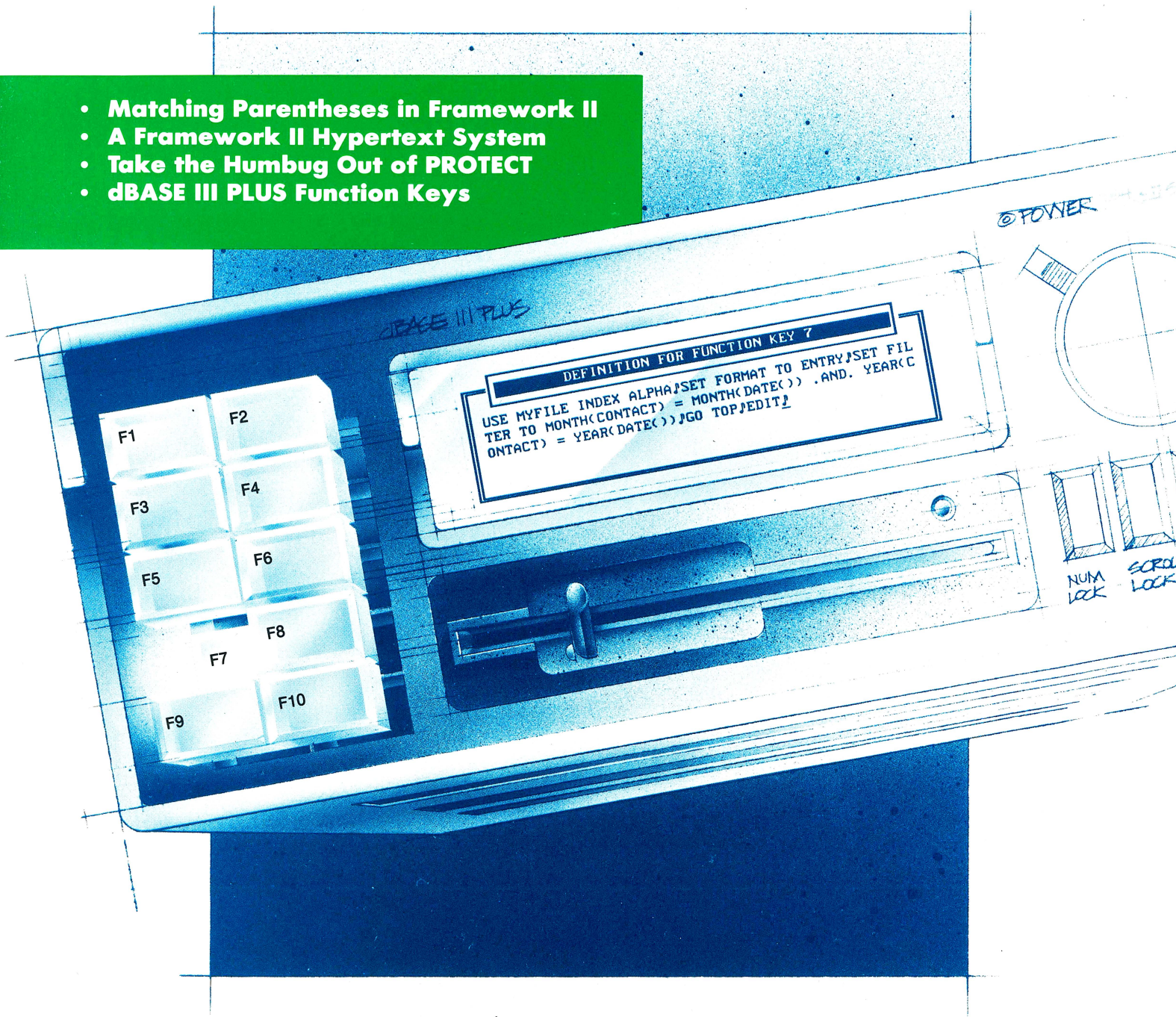
# TechNotes

ASHTON-TATE'S TECHNICAL JOURNAL
FEATURING dBASE AND FRAMEWORK

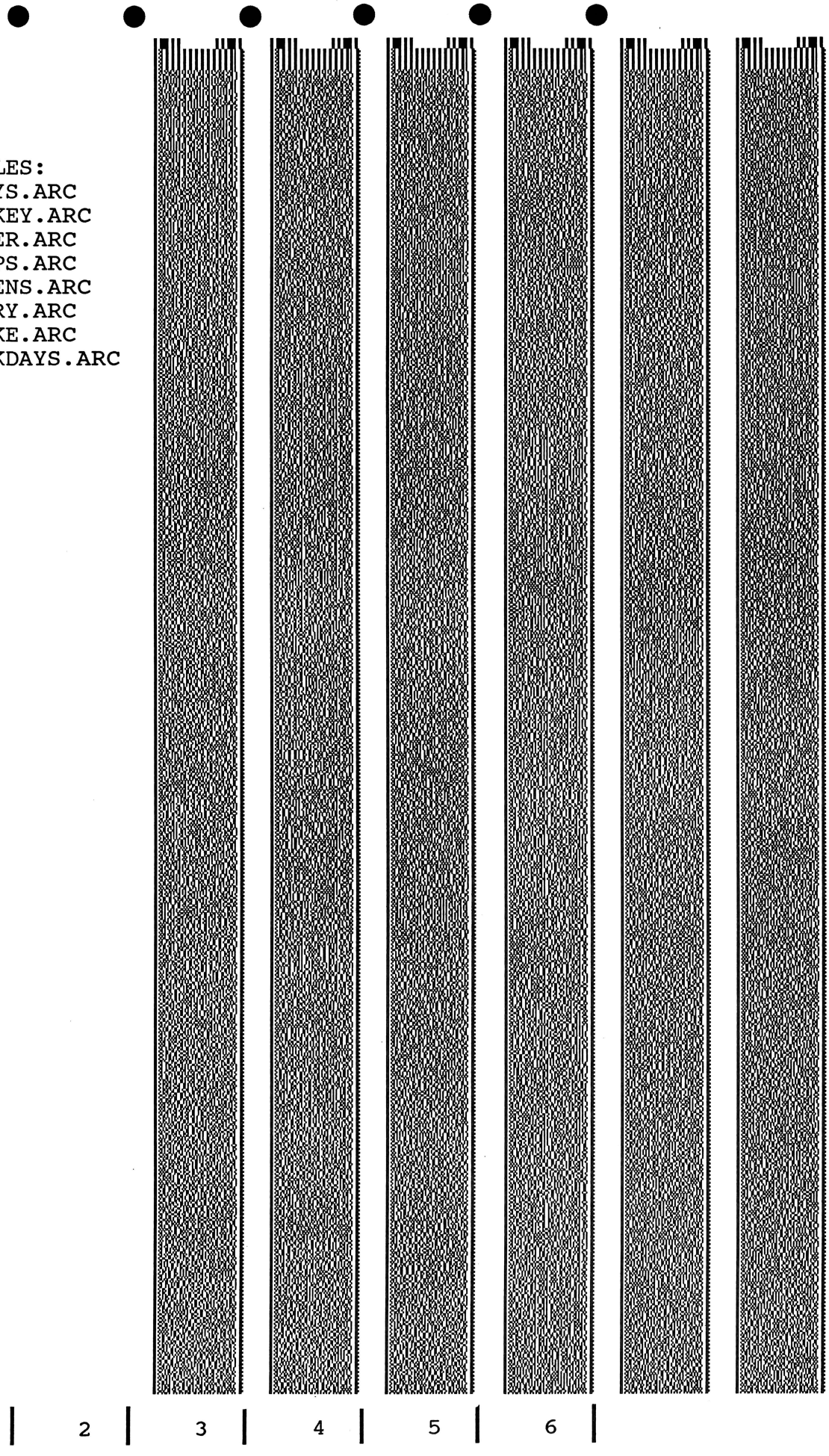
DECEMBER 1987
$6.00

- **Matching Parentheses in Framework II**
- **A Framework II Hypertext System**
- **Take the Humbug Out of PROTECT**
- **dBASE III PLUS Function Keys**




ASHTON-TATE

FILES:
FKEYS.ARC
GETKEY.ARC
HYPER.ARC
ICAPS.ARC
PARENS.ARC
RETRY.ARC
SNAKE.ARC
WEEKDAYS.ARC



# TechNotes

## TABLE OF CONTENTS

### DECEMBER 1987


| | | |
|---|---|---|
| **dBASE III PLUS** | Functioning Function Keys<br>*by Chuck Litzell* | 4 |
| | Wrapped Column Reports<br>*by Roger Wegehoft and Vincent Mendillo* | 13 |
| | Taking the Humbug Out of PROTECT<br>*by Richard Bunter and James Bradbury* | 22 |
| | Programmer's Casebook: Try RETRY Again<br>*by Kenneth N. Getz* | 27 |
| | dBASE III PLUS Tips | 35 |
| **Framework II** | A Hypertext System for Framework II<br>*by Roger Wegehoft* | 44 |
| | Finding Matching Parentheses<br>*by Quinn Wildman and Kenneth N. Getz* | 51 |
| | Framework II Tips | 58 |
| **Departments** | Read Me First<br>*by Brenda Johnson-Grau* | 3 |
| | Anomalies | 61 |
| | On-Line Services | 64 |
| | Letters | 67 |
| | Corrigenda | 69 |
| | Cumulative Index for 1987 | 70 |

TechNotes

**Publisher**
Ashton-Tate
Support Products Center

*Richard Goepel,* Manager

**Editor-in-Chief**
*Brenda Johnson-Grau*

**Editors**
*James Bradbury*
*Kenneth N. Getz*
*Charles Litzell*

**Production Editor**
*Tom Pasley*

**Technical Staff**
*Larry Abare, Rozita Afar, Richard Augelli, Rodney Barbati, Richard Bunter, Mike Carlisle, Fred Crownover, Rick Fillman, James Chuang, Peter Gold, Mina Gussin, Scott Haynes, J. Mike Hedblom, Clint Helton, Errol Jackson, Matt Johnson, Gail Kulikowsky, Steve Kurasch, Chris LoVasco, Jeffrey McCrimon, Jerry Mendes, Vincent Mendillo, Jeffrey Moore, Keith Mund, Joseph Perret, John Regen, Cathy Shikatani, Joseph Stolz, Naoma Stow, Philip Talsky, Mitra Toossi, Hiram Vega, Ralph Wakefield, Mark Ware, Roger Wegehoft, Quinn Wildman*

**Contributing Editors**
*Paul Bukauskas (International), Liz Burke (MultiMate), Steve Silverwood (On-Line Services)*

**Production Staff**
*Gary Ashcavai, D.A. Gary Bare, Tim Dunne, Darrell Evers, Patty Pesek, Ron Powell, Ann O. Taono*

**Product Development**
*Forefront Development Center: Marilyn McMahon, Greg Stikeleather*
*Glendale Development Center*

## DISCLAIMER NOTICE

The information contained in this publication is designed to help you develop and improve your programming skills and to assist you in using Ashton-Tate products. Ashton-Tate assumes no responsibility whatsoever for the uses made of any of the software code in this issue, whether modified or unmodified.

If you discover what you believe is an anomaly (as defined under **Anomalies**) not presented in this issue or others, please send a detailed description of the possible anomaly to:

Ashton-Tate Software Support Center
20101 Hamilton Avenue
Torrance, CA 90502-1319

If the reported anomaly can be reproduced by our technicians, it may be added to a future issue of ***TechNotes.***

The policies that are described in ***TechNotes*** apply only to the United States (and possessions), Canada, and Mexico. Residents of other foreign countries should contact a local Ashton-Tate representative in their country for policy information and technical support.

Ashton-Tate and the authors will not be liable for special, incidental, consequential, indirect or other similar damages, even if we have been advised of the possibility of such damages. This means we are not responsible for damages or costs incurred as a result of loss of time, loss of data, loss of profits or revenues, or loss of use of the software. In addition, we are not responsible for damages or costs incurred in connection with obtaining substitute software, claims by others, inconvenience or similar costs.

No warranties, either expressed or implied, are made regarding the contents of this publication, its merchantability or fitness for any particular purpose, all of which are hereby expressly disclaimed. Note, however, that some states do not allow the exclusion or limitation of warranties or of direct, incidental or consequential damages, so these warranty limitations may not apply to you.

In no event will our liability for any damages to you or any other person ever exceed the price paid for your subscription to ***TechNotes***, regardless of any form of the claim.

Additional statements by agents, employees, distributors and dealers of Ashton-Tate do not constitute warranties by Ashton-Tate and do not bind Ashton-Tate.

. . . . . .

***TechNotes*** (ISSN 0883-6493) is an Ashton-Tate Support Products Center publication containing technical reference information on Ashton-Tate products.

Entire contents Copyright © Ashton-Tate Corporation 1987. All rights reserved.

Commercial reproduction of the material appearing in ***TechNotes*** is forbidden without prior written permission. Direct requests for reproduction permission to Ashton-Tate Support Products Center, 20101 Hamilton Avenue, Torrance, CA 90502-1319, ATTN: Brenda Johnson-Grau.

***TechNotes*** is published monthly at a subscription price of $50.00 for one year. Add $15 for Canada and Mexico and add $25 for overseas. Back issues are available for $10.00 each (1985 issues) and $7.00 each (1986 and 1987 issues). ***TechNotes*** is published by Ashton-Tate, 20101 Hamilton Avenue, Torrance, CA 90502-1319.

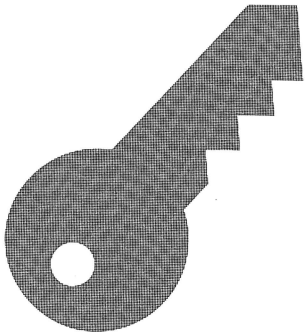
POSTMASTER: Send subscription inquiries and address changes to ***TechNotes*** Subscription Service, P.O. Box 3729, Escondido, CA 92025-0929.

**Subscription Inquiries**: Write ***TechNotes*** Customer Service, P.O. Box 3729, Escondido, CA 92025-0929. Call 800/828-2515 or 800/331-4164. From outside the U.S. call: 619/485-6535. When writing or calling, be sure that you have your current mailing label available. Your expiration date is listed on the far right of your mailing label.

dBASE, dBASE II, dBASE III, Ashton-Tate and Framework are registered trademarks and RapidFile, dBASE III PLUS, Framework II, RunTime, and FRED are trademarks of Ashton-Tate Corporation. Some of the trademarks of other companies mentioned in this issue appear for identification purposes only. MultiMate is a registered trademark and MultiMate Advantage, MultiMate GraphLink and the MultiMate Business Advantage Keyboard are trademarks of MultiMate International Corporation, an Ashton-Tate company. On-File is a trademark under license from Facts on File, Inc.

The Ashton-Tate Software Support Center's hours are 6:00 a.m. to 4:30 p.m., PST, Monday through Friday, except on Thursday when we close at 4:00 p.m. To reach our technicians for technical support, call (213) 329-0086. Be sure to have your software serial number ready for our operators.

# READ ME FIRST



By Brenda Johnson-Grau

"Functioning Function Keys," by Chuck Litzell, allows you to create complex function–key macros for dBASE III PLUS. It provides an excellent example of how to make programming decisions that optimize dBASE's performance, *and* it shows you how to get the most out of your function keys.

"Wrapped Column Reports," by Roger Wegehoft and Vincent Mendillo is a new way to handle snaked-column reports. It allows you to create dBASE III PLUS reports in both series side-by-side format and side-by-side series format (a subtle but useful distinction).

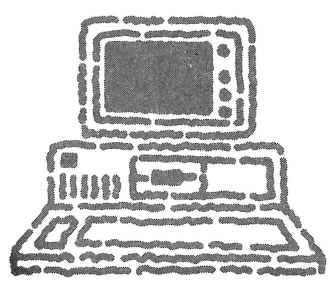
This month's LAN article is an explanation of how to use PROTECT and, in deference to the season, it has a holiday theme: Scrooge is setting up a multiuser system for Scrooge and Marley, Inc. "Taking the Humbug Out of PROTECT," by Richard Bunter and James Bradbury, provides a case study complete with tips for using PROTECT and setting rank and privilege in the dBASE multiuser environment.

As in many other block–structured languages, parentheses and curly braces are a major element in the FRED programming language. Every opening parenthesis or opening curly brace must be matched by a closing one. Careful indentation and good powers of concentration while counting parentheses or braces, can ensure everything matches up; but what if you do find a stray? Use the program in "Finding Matching Parentheses," by Quinn Wildman and Ken Getz.

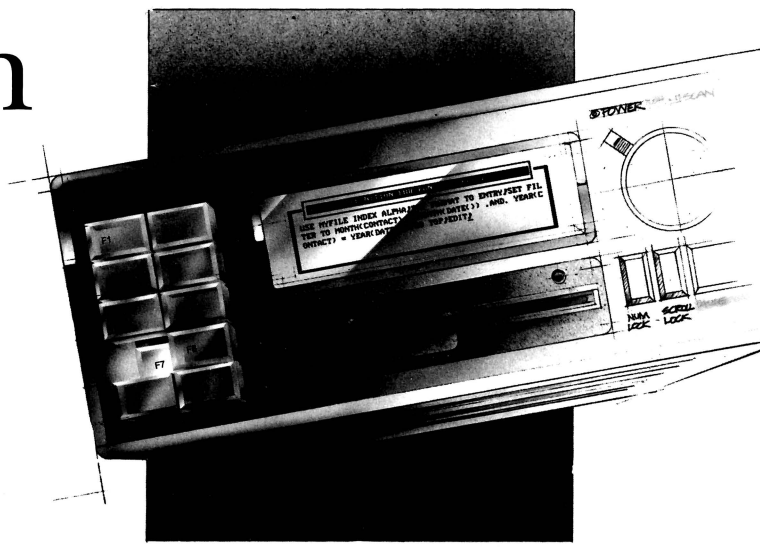
"A Hypertext System for Framework II," by Roger Wegehoft, uses FRED code and an idea developed by Theodor H. Nelson to create a sample hypertext system for Framework II. "Hypertext" is defined as a type of writing that branches upon request. In essence, it allows you to link text to particular keywords in a main document—making the main document, in effect, three-dimensional.

Finally, the cumulative index is back. We're trying to make the index as useful as possible so that you can easily research a question or a problem. Please let us know if you would like to see any changes in how we present the index.

Happy Holidays!



# Functioning Function Keys



By Chuck Litzell

Fkeydef.PRG is a dBASE III PLUS program that lets you define and edit strings attached to function keys. The SET FUNCTION command permits assignment of values to function keys 2 through 10. When a function key is pressed, dBASE plays out the assigned string as if you had pressed the keys. Because it's reserved for HELP, **F1** may not be reassigned.

Fkeydef lets you assign any character in the full extended-ASCII range to a function key. This includes all the printable characters, such as letters, numbers, and the graphics characters in the IBM extended graphics character set. Control keys are also valid, and this can lead to some interesting uses. For example, you might have a PHONE_NO field in a database file where phone numbers are typed in the format:

```
(555) 989-9312
```

To save keystrokes you could redefine **F2** with

```
SET FUNCTION 2 TO REPLICATE(CHR(19), 10) + CHR(22) + "(" +;
                  REPLICATE(CHR(4),   3) + ") "    + ;
                  REPLICATE(CHR(4),   3) + "-"     +;
                  REPLICATE(CHR(4),   4) + CHR(22) + CHR(13)
```

Thereafter, you would be able to enter that same phone number as

```
5559899312<F2>
```

What does that long string you assigned to **F2** do? CHR(19) is the value dBASE assigns to the **Leftarrow** key, and CHR(4) is the **Rightarrow** key. CHR(22) is the **Ins** key, and CHR(13) is the **Return** key. When you fill in the phone number, and press **F2** as above, dBASE presses the **Leftarrow** ten times, toggles **Ins**, types a left parenthesis, presses **Rightarrow** three times, types a right parenthesis and a space, presses **Rightarrow** three more times, types a dash, presses **Rightarrow** four times, toggles **Ins,** and presses **Return**. What formerly took 24 keystrokes has now been reduced to one.

All the keys that dBASE III PLUS recognizes have a CHR() value, with the exception of function keys 2 through 10. The value that dBASE assigns to a key is the same value returned by the INKEY() func-

tion. Most of these are listed in a chart in the documention for INKEY() in the "Functions" section of *Learning and Using dBASE III PLUS.*

The special keys (the gray keys on an IBM keyboard) have INKEY() values in the range 1 to 31. Each one has an alternative keystroke. For example, **Ctrl-E** is the equivalent of **Uparrow,** and **Ctrl-M** is the same as **Return**. The mapping of control keys to the special keys follows historical precedent, as users of dBASE II and WordStar® will recall.

Some of the values in the 1 to 31 range have no key assignments in the chart because dBASE assigns no special meaning to them. INKEY() returns 9 for the **Tab** key, for example, but since dBASE doesn't use the **Tab** key, it isn't in the chart.

Function keys are a special case. INKEY() returns a value of 28 for **F1** and negative values for **F2** through **F10**. We can test for a function key with INKEY(). But, since a negative value as the argument of the CHR() function is an error, **F1** is the only function key that can be used within a string assigned to another function key.

Fkeydef must recognize all values that INKEY() can possibly return. Table 2 is an abridged version of the table in the INKEY() documentation, including the undocumented values for control and function keys.

## Using Fkeydef

Fkeydef is a stand–alone program that can be run by typing

```
DO Fkeydef
```

at the dot prompt. Fkeydef first looks for a .MEM file, Fkeys.MEM. This file is used to store the strings assigned to the nine assignable function keys. If Fkeys.MEM exists, the current values are RESTOREd. If it doesn't exist, it is created with nine empty strings.

Fkeydef requests that the user press the function key to be redefined, then displays the current string assigned to the key that was pressed. Pressing any key on the keyboard will record the key in the string at the cursor position. Exceptions are the function keys and the **Backspace** key. These keys perform editing functions, such as moving the cursor and inserting or deleting characters.

**Ctrl-L** is another special case. **Ctrl-L** enters the value for the **Leftarrow** rather than its true value, CHR(12). The value dBASE assigns to the **Left arrow** is CHR(19), or **Ctrl-S.** The BIOS interprets this key as a pause character. It stops the computer until **Ctrl-Q** is pressed. The **Ctrl-Q** and **Ctrl-S** characters are labeled XON and XOFF in ASCII tables, the mnemonic codes for Transmit On and Transmit Off. Frequently, the computer interprets CHR(19) as XOFF before the INKEY() function intercepts it. **Ctrl-L** was selected as an alternative to the **Leftarrow** both for its mnemonic association with "left" and because it is of little use as a keystroke in dBASE III PLUS.

Table 1 shows keys having special functions in Fkeydef.

## How the Program Works

Implementing a full-screen editor of any kind in an interpreted language like dBASE III PLUS is an ambitious project. If the program is too slow, it won't be used. Optimization and compromise are

| Key | Editing Function |
|---|---|
| F1 | Displays the name of the key corresponding to the character at the cursor position. |
| F2 | Ends editing session. |
| F3 | Deletes character at cursor position. |
| F4 | Inserts a space at the cursor position. |
| F5 | Moves cursor to beginning of string. |
| F6 | Moves cursor to end of string. |
| F7 | Moves cursor one character to left. |
| F8 | Moves cursor one character to right. |
| F9 | Moves cursor five characters to left. |
| F10 | Moves cursor five characters to right. |
| Backspace | Deletes character to the left of the cursor. |
| Ctrl-L | Enters a Leftarrow at the cursor position. |

Table 1

Revised return values of INKEY()

| Special Keys | Alternative Keys | Inkey Return Value |
|---|---|---|
| F10 | | -9 |
| F9 | | -8 |
| F8 | | -7 |
| F7 | | -6 |
| F6 | | -5 |
| F5 | | -4 |
| F4 | | -3 |
| F3 | | -2 |
| F2 | | -1 |
| Home | Ctrl-A | 1 |
| Ctrl-Right | Ctrl-B | 2 |
| PgDn | Ctrl-C, Ctrl-@ | 3 |
| Right | Ctrl-D | 4 |
| Up | Ctrl-E | 5 |
| End | Ctrl-F | 6 |
| Del | Ctrl-G | 7 |
| | Ctrl-H | 8 |
| Tab | Ctrl-I | 9 |
| Ctrl-Return | Ctrl-J | 10 |
| | Ctrl-K | 11 |
| | Ctrl-L | 12 |
| Return | Ctrl-M | 13 |
| | Ctrl-N | 14 |
| Shift-Tab | Ctrl-O | 15 |
| | Ctrl-P | 16 |
| | Ctrl-Q | 17 |
| PgUp | Ctrl-R | 18 |
| Left | Ctrl-S | 19 |
| | Ctrl-T | 20 |
| | Ctrl-U | 21 |
| Ins | Ctrl-V | 22 |
| Ctrl-End | Ctrl-W | 23 |
| Down | Ctrl-X | 24 |
| | Ctrl-Y | 25 |
| Ctrl-Left | Ctrl-Z | 26 |
| Escape | Ctrl-[ | 27 |
| F1 | Ctrl-\ | 28 |
| Ctrl-Home | Ctrl-] | 29 |
| Ctrl-PgDn | Ctrl-^, Ctrl-Del | 30 |
| Ctrl-PgUp | Ctrl-_ | 31 |
| Backspace | | 127 |

Table 2

the solutions. On a standard IBM PC, Fkeydef drags noticeably, but not intolerably. The maximum length of a string to be edited is about 230 characters, allowing for the rest of the SET FUNCTION command. Most strings will be much smaller. Considering, too, that the program will not be used as heavily as, say, a data-entry program, we have an opportunity to favor functionality over execution speed. If the program's task were any larger, or the planned use was greater, it would be necessary to use a much different program design.

The program was compromised in several ways. It would have been nice to display keynames instead of IBM's graphic characters for control codes. The program statements necessary to accomplish this, however, slowed the program to the point where it was unusable. The **F1** function is the compromise. Another abandoned feature was a counter at the bottom of the screen that continuously displayed the length of the string. The increase in speed outweighed the utility of that feature.

The initial version of Fkeydef had a DO CASE structure in the central DO WHILE loop with a CASE for each of the eleven editing keys. The OTHERWISE case handled all the other keys. To optimize the program, it was necessary to minimize the number of program lines in the DO WHILE loop, since the program's response is directly tied to how fast dBASE can get through the loop. In the current version, the DO WHILE does only what's necessary. It puts the cursor where it needs to be, then gets a key from the user. The key is tested to see whether it belongs to the *class* of editing keys. If it does, the program calls a procedure, Comkey, to handle the key. If it isn't an edit key, the new character is installed in the string, and the cursor's position in the string is updated.

The routines moved to the Comkey procedure were optimized, too. First, several of the function keys do nothing but update the cursor position. These were combined into one CASE of the DO CASE structure. The program uses a table to translate the function key into a value that is used to adjust the cursor position.

The updated cursor position must be checked so that it doesn't point before the start of the string or past the end. This test was moved to the main DO WHILE loop so that it is done in one line of code, and only once for every pass through the loop.

Any of the CASES in the DO CASE structure that have accomplished their task do an immediate RETURN; so, dBASE doesn't examine the remaining cases before returning to the main loop. The CASE that requires the fastest response, moving the cursor, was moved to the top of the DO CASE to minimize the number of lines interpreted. The longest case, which displays the keyname, is last so that dBASE doesn't have to read through it to get to the other cases.

Three of the edit keys, **F3**, **F4,** and **Backspace**, change the string in such a way that it must be redisplayed. The code to redisplay the string was placed after the ENDCASE in Comkey so that it is not repeated three times. The CASEs that don't need to redisplay the string will never reach this routine, since they RETURN rather than fall through the ENDCASE.

There are several things we didn't do to optimize the program. First, comments, leading spaces, and blank lines, which take time to process, could be omitted. Memory variable names could be shortened, and literals could be used in place of memory variables for constants. All dBASE commands and functions could be abbreviated to four letters. All these actions would help speed a program's execution simply because dBASE would be processing fewer bytes. That we didn't do any of these things makes the published program more readable, but you might decide to implement them as you enter the program.

## Resetting the Function Keys

Fkeydef resets the function key when you finish editing and tells the program to save the new definition. However, when you next come back to dBASE, the function keys will default to the strings preassigned by dBASE or to definitions established in Config.DB.

Fkeyset.PRG is a short program that reads the strings stored in Fkeys.MEM and reassigns the function keys. If the string in Fkeys.MEM is empty, no reassignment is made. To run Fkeyset, type

```
DO Fkeyset
```

at the dot prompt. You may wish to add this line to your start-up program, if you have one. Alternatively, you could add

```
COMMAND = DO Fkeyset
```

to Config.DB using MODIFY COMMAND or another plain text editor. Make sure, too, there is only one "COMMAND =" line in Config.DB. If you work with ASSIST, you could add

```
ASSIST
```

as the line immediately before the RETURN statement at the end of Fkeyset.

## Fkeydef.PRG

```
* Program ...: Fkeydef.PRG
* Author ....: Chuck Litzell
* Date ......: December 1, 1987
* Version ...: dBASE III PLUS, versions 1.0 and 1.1.
* Notes .....: Define and edit strings assigned to function keys 2
*              through 10.
*
*              Definitions are stored in a .MEM file -- Fkeys.MEM.
*              Use Fkeyset.PRG to reset function keys when dBASE
*              is reloaded.
*
*
SET TALK OFF
SET ESCAPE OFF
SET PROCEDURE TO Fkeydef

* ---Table of keynames.
*          0123456789 123456789 123456789 123456789 123456789
keytab1  = "HOME      CTL-RIGHT PGDN      RIGHT     UP        " +;
           "END       DELETE    CTL-H     TAB       CTL-RETURN"
keytab2  = "CTL-K     CTL-L     RETURN    CTL-N     SHIFT-TAB " +;
           "CTL-P     CTL-Q     PGUP      LEFT      CTL-T     "
keytab3  = "CTL-U     INS       CTL-END   DOWN      CTL-Y     " +;
           "CTL-LEFT  ESCAPE    F1        CTL-HOME  CTL-PGDN  "
keytab4  = "CTL-PGUP  SPACE"

* ---Miscellaneous constants.
bar      = CHR(186)
bell     = CHR(7)
nulstr   = ""
blank    = " "
* ---Values returned by INKEY() for function keys and backspace.
bkspkey  = 127
f1key    =  28
f2key    =  -1
f3key    =  -2
f4key    =  -3
f5key    =  -4
f6key    =  -5
f7key    =  -6
f8key    =  -7
f9key    =  -8
f10key   =  -9

* ---Status line.
statline = "  F2 DONE" + bar +;
           "F3 DEL" + bar +;
           "F4 INS" + bar +;
           "F5 HOME" + bar +;
           "F6 END" + bar +;
           "F7 LEFT" + bar +;
           "F8 RIGHT" + bar +;
           "F9 LEFT 5"+ bar +;
           "F10 RIGHT 5    "
helpmsg  = " Press F1 to Identify Symbol "

* ---Restore current key definitions.
```

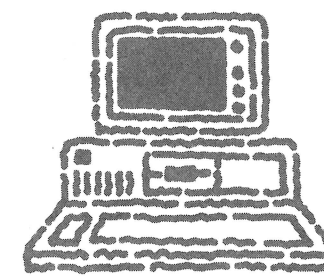
```
IF .NOT. FILE("Fkeys.MEM")
   STORE nulstr TO fk_2, fk_3, fk_4, fk_5, fk_6, fk_7, fk_8, fk_9, fk_10
   SAVE ALL like fk_* TO Fkeys.MEM
ENDIF
RESTORE FROM Fkeys.MEM ADDITIVE


DO WHILE .T.

   * ---Draw Screen.
   CLEAR
   @  3,  0 TO  3, 79 DOUBLE
   @  3, 24 SAY " EDIT FUNCTION KEY DEFINITION "
   @  6, 13 TO 13, 66 DOUBLE

   * ---Show status line and help message in reverse video.
   * ---The GET/CLEAR GETS combination does this without
   * ---altering the current color settings.
   @ 21,  0 GET statline
   CLEAR GETS

   * ---What function key to define?
   @ 23, 20 SAY "Press Function Key to Define, F1 to Quit"
   fkey = 0
   DO WHILE fkey > -1 .OR. fkey < -9
      fkey = INKEY()
      IF fkey = f1key
         EXIT
      ENDIF
   ENDDO
   IF fkey = f1key
      EXIT
   ENDIF
   fkey    = LTRIM(STR(ABS(fkey) + 1))
   keymac  = "FK_" + fkey
   keydef  = &keymac
   fkeymsg = SPACE(8) + "DEFINITION FOR FUNCTION KEY " +  fkey + SPACE(10 - LEN(fkey))

   @ 23,  0 CLEAR TO 24, 79
   @  6, 17 GET fkeymsg
   @  5, 16 TO  7, 63
   @ 15, 24 TO 18, 54
   @ 17, 27 TO 19, 51 DOUBLE
   @ 18, 28 CLEAR TO 18, 50
   @ 16, 25 GET helpmsg
   CLEAR GETS

   * ---Display current key definition.
   ln = 0
   DO WHILE ln <= LEN(keydef) / 50
      @  8 + ln, 15 SAY LEFT(SUBSTR(keydef + blank, ln * 50 + 1, 50) + SPACE(50), 50)
      ln = ln + 1
   ENDDO

   ptr   = 1
   done  = .F.
   DO WHILE .NOT. done

      * ---Set cursor for input.
      @ INT((ptr - 1) / 50) + 8, 15 + MOD(ptr - 1, 50) SAY nulstr
```

```
      * ---Get a key.
      press = 0
      DO WHILE press = 0
        press = INKEY()
      ENDDO

      * ---Is it an edit key?
      IF press < 0 .OR. press = f1key .OR. press = bkspkey
        DO comkey
      ELSE
        * ---Kludge to make Ctrl-L act like Left-Arrow.
        IF press = 12
          press = 19
        ENDIF

        * ---Install character and display it.
        * ---Note: $ is an undocumented synonymn of the
        *          ROW() and COL() functions.
        keydef = STUFF(keydef, ptr, 1, CHR(press))
        @ $, $ SAY SUBSTR(keydef + blank, ptr, 1)
        * ---Point to next position.
        ptr = ptr + 1
      ENDIF

      * ---Edit and cursor positioning routines adjust ptr but don't
      * ---care if its within the valid range.  That is taken care of
      * ---right here.
      ptr = MAX(MIN(ptr, LEN(keydef) + 1), 1)
    ENDDO

    * ---Save definition, if desired, and exit.
    @ 22, 0 CLEAR
    saveit = .T.
    @ 23, 25 SAY "Save New Definition for F&fkey.?" GET saveit PICTURE "Y"
    READ
    @ 23,  0 CLEAR
    IF saveit
      SET FUNCTION &fkey TO keydef
      STORE keydef TO &keymac
    ENDIF
  ENDDO

  SET SAFETY OFF
  SAVE ALL LIKE FK_* TO Fkeys.MEM
  SET SAFETY ON
  CLOSE PROCEDURE
  CLEAR
  RETURN
  * EOP Keydef.

  PROCEDURE Comkey
  * ---Process key pressed.
  DO CASE

  CASE press <= f5key .AND. press >= f10key
    * ---This key just modifies ptr.
    ptr = ptr + VAL(SUBSTR("0005-0050001-0010254-254", (press + 9) * 4 + 1, 4))
    RETURN
```

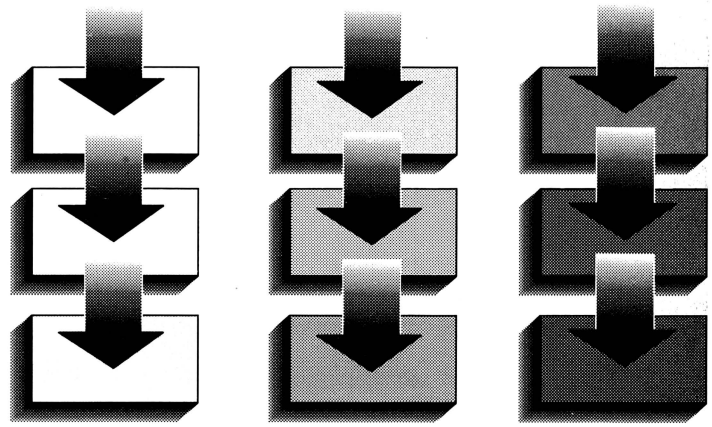
```
CASE press = f2key
* ---Request to end edit.
   @ 23, 31 SAY "Finished Editing?" GET done PICTURE "Y"
   READ
   @ 23, 25 CLEAR
   RETURN

CASE press = f3key
   * ---Delete character at ptr.
   IF LEN(keydef) >= ptr
      keydef = STUFF(keydef, ptr, 1, nulstr)
   ENDIF

CASE press = f4key
   * ---Insert a blank at ptr.
   keydef = STUFF(keydef, ptr, 0, blank)

CASE press = bkspkey
   * ---Destructive backspace.
   IF LEN(keydef) + 1 >= ptr .AND. ptr > 1
      keydef = STUFF(keydef, ptr - 1, 1, nulstr)
      ptr = ptr - 1
   ENDIF

CASE press = f1key
   * ---Display keyname of character at ptr.
   IF ptr > LEN(keydef)
      interp = "<End of Definition>"
   ELSE
      ch = ASC(SUBSTR(keydef + blank, ptr, 1))
      IF ch < 33
         keylin = LTRIM(STR(INT((ch - 1) / 10) + 1))
         interp = LTRIM(SUBSTR(keytab&keylin, MOD(ch - 1, 10) * 10 + 1, 10))
      ELSE
         interp = "CHR(" + LTRIM(STR(ch)) + ")"
      ENDIF
      interp = ["] + CHR(ch) + [" - ] + interp
   ENDIF
   @ 18, 28 CLEAR TO 18, 50
   @ 18, 40 - LEN(TRIM(interp)) / 2 SAY interp
   RETURN
ENDCASE

* ---Redisplay definition. This routine is executed only by
* ---F3, F4, or Backspace, since they may alter the definition
* ---string in the middle.
ln = 0
DO WHILE ln < INT(LEN(keydef) / 50) + 1
   @  8 + ln, 15 SAY SUBSTR(keydef + blank, (ln * 50) + 1, 50) + blank
   ln = ln + 1
ENDDO
RETURN
* EOP Comkey.
```

## Fkeyset.PRG

```
* Program ...: Fkeyset.PRG
* Author ....: Chuck Litzell
* Date ......: December 1, 1987
* Version ...: dBASE III PLUS, versions 1.0 and 1.1.
* Notes .....: Set function keys to strings stored in Fkeys.MEM.
*
*              Use Fkeydef.PRG to define and edit the strings in
*              Fkeys.MEM.
SET TALK OFF

IF .NOT. FILE("Fkeys.MEM")
   ? "Can't open Fkeys.MEM.  Function Keys not Reset."
   RETURN
ENDIF
RESTORE FROM Fkeys ADDITIVE
counter = "2"
DO WHILE VAL(counter) < 11
   keydef = FK_&counter
   IF "" # TRIM(keydef)
      SET FUNCTION &counter TO keydef
   ENDIF
   counter = LTRIM(STR(VAL(counter)+1))
ENDDO
? "Function Keys have been Reset"
RETURN
* EOP Fkeyset.PRG.
```



# Wrapped Column Reports



By Roger Wegehoft and Vincent Mendillo

An articled titled "Snaked Column Reporting" in the June 1987 ***TechNotes*** described a dBASE III PLUS program that created reports where the data wrapped from the bottom of one column to the top of the next. Expanding on that, this article allows you to generate the same type of reports but with two presentation modes, direct support for all data types, random field order, variable column grouping, a simple menu-driven interface, and output using the built-in dBASE III PLUS report form generator.

## About the Program

As explained in the previous article, snaked-column reporting refers to the way data is sequenced and formatted on the page. The programs in this article enable you to generate wrapped column reports in two basic styles. The first style is called "Series Side-by-Side" or "SRSXS." In this mode, which is illustrated in Figure 1, the series of data fields in the source database file indicated by the capital letters A, B, and C are allowed to extend downward in their respective columns. When the data reaches the bottom of a defined page, or block as they are called here, it resumes at the top of the adjacent block. A block can contain up to F fields, where F is given by the expression

F = 24/B

and B is the number of blocks specified in the report setup. The constant 24 in the formula stems from the maximum number of columns allowed in a dBASE III PLUS report using the built-in report

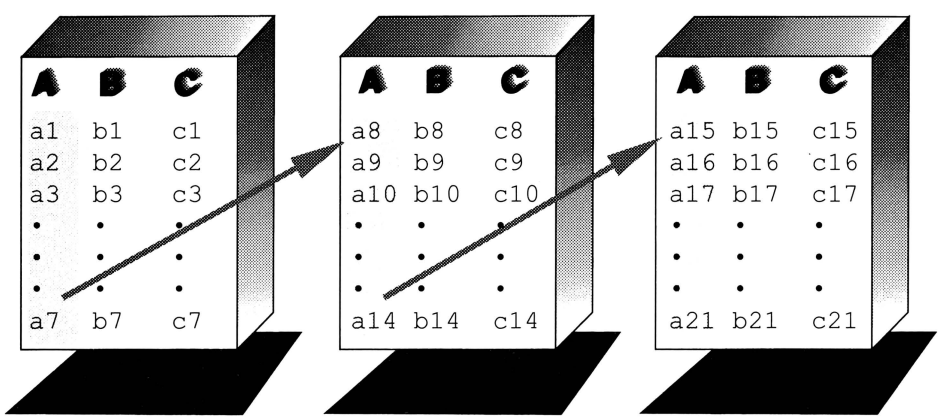



Figure 1: Series Side–by–Side

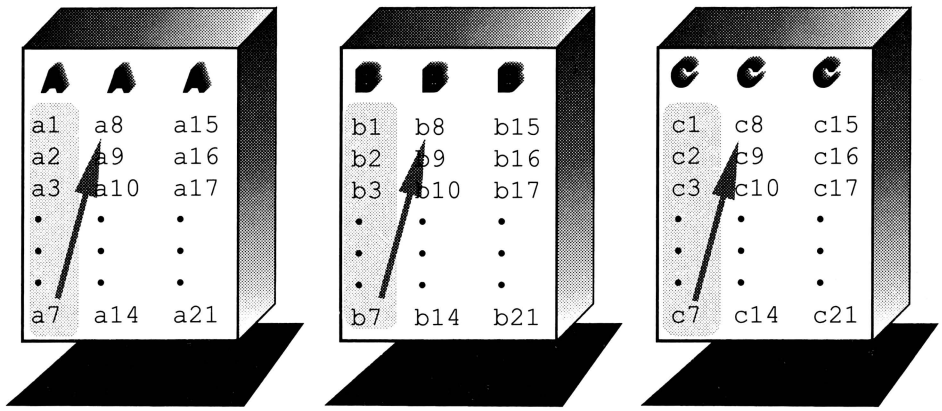



Figure 2: Side–by–Side Series

generator. This limitation can be overcome by concatenating fields up to the 254-character line limit.

The second style is called "Side-by-Side Series" or "SXSSR" and is shown in Figure 2. This mode wraps data from the same field in adjacent columns within a given block. Each block contains a different field. Here again the previous equation for the maximum number of fields per block applies.

With each style you can select the number and order of the source database fields that will appear in the formatted destination database. Also note that no transformation or coercion to a particular data type is required as the program preserves the data type from input to output.

## Usage

The program can be divided into three logical parts. The first part deals with prompting and collection of the names of the source and destination databases as well as the values for the style or mode, block count, lines per page, and the number of lines to allow for the report header. The second part is concerned with the composition and structure of the destination database. The third part actually moves the data.

During data entry we assume that all program modules, as well as the source and destination database files, lie within the current default directory and that database files have the default .DBF extension which is standard for dBASE III PLUS.

To start the program, type the following command at the dot prompt:

```
DO Snake
```

Answer all prompts by entering your own information or by pressing **Return** and accepting the default value. After all prompts have been answered, the program will inform you from time to time of its progress. Be forewarned, because dBASE is an interpreted language and because there are many nested DO WHILE loops and REPLACE statements, this program is no speed demon; formatting for a large database could take a long time.

In the destination database file you may find some very odd–looking field names. These can be explained with the following format:

BnSnDnXXXX

Where Bn is the Block number, Sn is the source database file field number, and Dn is the destination database file field number. XXXX will always be the first four letters of the source database file structure and is appended as an aid to interpreting the output of the destination database file structure.

After the program has completed formatting the destination database file, you can invoke the dBASE III PLUS report form generator and create titles, column headings, and so on.

## Snake.PRG

```
* Program..:  Snake.PRG
* Authors..:  Vincent Mendillo and Roger Wegehoft
* Date.....:  December 1, 1987
* Version..:  dBASE III PLUS
* Notes....:  Snake.PRG is the main program for producing snaked reports.
*
* ---Configure working environment.
CLEAR
CLOSE ALL
SET BELL OFF
SET CONFIRM ON
SET ECHO OFF
SET MENU OFF
SET SAFETY OFF
SET STATUS ON
SET TALK OFF
*
mdst_dbf     = SPACE(65)                    && Formatted output database file.
msrc_struc= "Struct"
mmode      = 0
srsxs      = 1
sxssr      = 2
DO WHILE .T.
   CLEAR
   msrc_dbf = SPACE(65)                   && Original database file.
   SET MESSAGE TO "Press <ESC> to Quit."
   msg = "*** Snaked-Column Report Generator ***"
   @  3, 40 - LEN(msg)/2 SAY msg
   @  5,  5 SAY "Source DBF (Filename only): ";
            GET msrc_dbf;
            PICTURE "@!S25"
   READ
   IF READKEY() = 12
      EXIT
   ENDIF
   msrc_dbf = LTRIM(TRIM(msrc_dbf))
   IF .NOT. FILE("&msrc_dbf..DBF")
      ?? CHR(7)
      @ 20,  0
      @ 20,  0 SAY "ERROR: The file &msrc_dbf..DBF does not exist.  " +;
                   "Please try again."
      WAIT
      LOOP
   ELSE
      *
      * ---Get name for final database file.
      * ---The default filename is "Outfile.DBF".
      @  7,  5 SAY "Output DBF (Filename only): ";
               GET mdst_dbf;
               PICTURE "@!S25"
      READ
      IF READKEY() = 12
         EXIT
      ENDIF
      IF LEN(TRIM(mdst_dbf)) = 0
         STORE "OUTFILE " TO mdst_dbf
         @  7,  33 GET mdst_dbf
```

```
   CLEAR GETS
ENDIF
mdst_dbf = LTRIM(TRIM(mdst_dbf))
*
* ---Make series selection.
mmode = 1
@  9,  5 SAY "Series Side-by-Side (1)"
@ 10,  5 SAY "Side-by-Side Series (2)"
@ 10, 52 GET mmode;
         PICTURE "9";
         RANGE 1, 2
READ
IF READKEY() = 12
   EXIT
ENDIF
*
* ---Block repeat count.
STORE 2 TO mblock
@ 12,  5 SAY "Block repeat count (between 2 and 12)"
@ 12, 51 GET mblock;
         PICTURE "99";
         RANGE 2, 12
READ
IF READKEY() = 12
   EXIT
ENDIF
*
* ---Page length calculation.
STORE 66 TO e_line
@ 15,  5 SAY "Number of lines per page (2 - 500)"
@ 15, 50 GET e_line;
        PICTURE "999";
        RANGE 2, 500
READ
IF READKEY() = 12
   EXIT
ENDIF
*
* ---Report header calculation is only performed if the page length
* ---is greater than 2.
STORE 0 TO e_head
IF e_line > 2
   maxhead = MIN(e_line - 2, 6)
   @ 16,  5 SAY "Number of lines in the report header (0 - " +;
              STR(maxhead, 1) + ")"
   @ 16, 52 GET e_head;
            PICTURE "9";
            RANGE 0, maxhead
   READ
   IF READKEY() = 12
      EXIT
   ENDIF
ENDIF
mpage_len = e_line - e_head
*
* ---Field order setup.
msg = "Processing setup.  Please wait."
@ 20,  0
@ 20,  0 SAY msg
USE &msrc_dbf
```

```
COPY STRUCTURE EXTENDED TO Strufile
USE Strufile
COPY STRUCTURE EXTENDED TO Xstrufile
USE Xstrufile
APPEND BLANK
REPLACE Field_name WITH "Field_ordr",;
     Field_type WITH "N",;
     Field_len  WITH 3,;
     Field_dec  WITH 0
APPEND BLANK
REPLACE Field_name WITH "Field_cols",;
     Field_type WITH "N",;
     Field_len  WITH 3,;
     Field_dec  WITH 0
CREATE Tstruct FROM Xstrufile
USE Tstruct
APPEND FROM Strufile

CLEAR
? "Please edit the source database file fields (coming up next)."
? "Enter a number next to each field name indicating the order"
? "in which that field should appear in the output destination"
? "database.  A zero or blank in the order field suppresses output "
? "of that field."
?
? "Press Ctrl-End when finished with the next screen to continue,"
? "   or press Esc when finished to quit."
?
?
WAIT
*
* ---Field order arrangement.
GO TOP
SET FIELDS TO Field_name, Field_ordr
BROWSE NOAPPEND FREEZE Field_ordr
IF READKEY() = 12
  EXIT
ENDIF
INDEX ON Field_ordr TO Tempndx
SET FIELDS OFF
*
* ---Create ordered database.
COUNT TO flds FOR Field_ordr > 0
IF flds * mblock > 128
  ?? CHR(7)
  @ 20,  0
  @ 20,  0 SAY "ERROR: Your output file will contain more than 128 fields.  "
  @ 21,  0   SAY "Program halting."
  WAIT
  EXIT
ENDIF
COPY TO Struct FOR Field_ordr > 0
*
? "Creating destination database..."
DO Snakmake
*
? "Arranging data..."
DO Snakgen
*
@ 20,  0
```

```
      @ 20,  0 SAY "Arrangement complete."
   ENDIF

   * ---Clean up working environment.
   ERASE Strufile.DBF
   ERASE Xstrufile.DBF
   ERASE Tstruct.DBF
   ERASE Struct.DBF
   ERASE Tempndx.NDX
   ERASE Dst_struc.DBF
   CLOSE ALL
   EXIT
ENDDO
*
* ---Restore environment.
SET BELL ON
SET CONFIRM OFF
SET MENU ON
SET MESSAGE TO
SET SAFETY ON
SET STATUS ON
SET TALK ON
RETURN
* ---EOP Snake.PRG.
```

## Snakmake.PRG

```
* Program..:  Snakmake.PRG
* Authors..:  Vincent Mendillo and Roger Wegehoft
* Date.....:  December 1, 1987
* Version..:  dBASE III PLUS
* Notes....:  Snakmake.PRG creates the output database for the snaked report.
*
STORE 0 TO mparm1, mparm2, block, source, dest
CLOSE ALL
SELECT 1
USE &msrc_struc
str1 = STR(mmode, 1, 0)                    && str1 = "1" or "2"
str2 = STR(3 - mmode, 1, 0)                && str2 = IIF(str1 = '1','2','1')
mparm&str1 = mblock
mparm&str2 = RECCOUNT()
COPY STRUCTURE TO Dst_struc
SELECT 2
USE Dst_struc
SELECT 1
GO TOP
DO WHILE (block < mparm1) .AND. .NOT. EOF()
   block  = block + 1
   source = 1
   DO WHILE (source <= mparm2)
      SELECT 1
      dest = dest + 1
      mfield_name = SUBSTR("B" + LTRIM(STR(block)) + ;
                           "S" + LTRIM(STR(source)) + ;
                           "D" + LTRIM(STR(dest)) + TRIM(field_name), 1, 10)
      mfield_type = field_type
      mfield_len  = field_len
      mfield_dec  = field_dec
      mfield_ordr = field_ordr
      mfield_cols = field_cols
```

```
      IF mmode = srsxs
         SKIP
      ENDIF
      SELECT 2
      APPEND BLANK
      REPLACE Field_name WITH mfield_name,;
              Field_type WITH mfield_type,;
              Field_len  WITH mfield_len,;
              Field_dec  WITH mfield_dec,;
              Field_ordr WITH mfield_ordr,;
              Field_cols WITH mfield_cols
      source = source + 1
   ENDDO
   SELECT 1
   IF mmode = srsxs
      GO TOP
   ELSE
      SKIP
   ENDIF
ENDDO
SELECT 2
CREATE &mdst_dbf FROM Dst_struc
ERASE Dst_struc.DBF
RETURN
* ---EOP Snakmake.PRG.
```

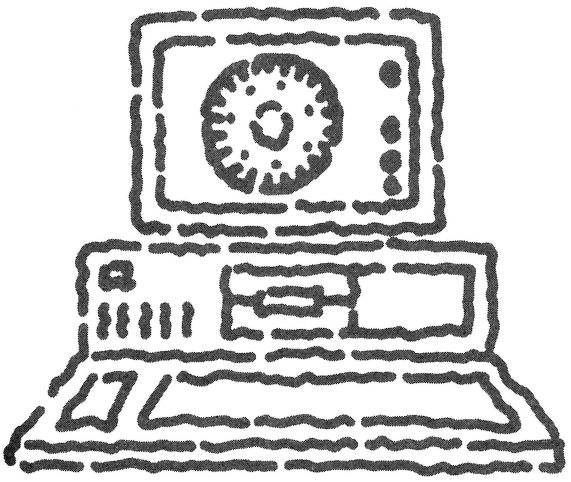
## Snakgen.PRG

```
* Program..:  Snakgen.PRG
* Authors..:  Vincent Mendillo and Roger Wegehoft
* Date.....:  December 1, 1987
* Version..:  dBASE III PLUS
* Notes....:  Snakgen.PRG generates the output database for the snaked
*             report.
*
CLOSE ALL
SELECT 1                                  && Source structure.
USE &msrc_struc
str1         = STR(mmode, 1, 0)
str2         = STR(3 - mmode, 1, 0)
local&str1= mblock
local&str2= RECCOUNT()
? "You chose " + STR(local1 * local2,3,1) + " output fields."
SELECT 3                                  && Destination database file.
USE &mdst_dbf
ZAP

SELECT 2                                  && Source database file.
USE &msrc_dbf
numrecs = RECCOUNT()
recnum  = 0
row     = ROW() + 2
@ row,  0 SAY "Currently working with source record number: "
col     = COL()
mlincnt = 1
DO WHILE .NOT. EOF()
   *
   * ---Set the current source base pointer.
   mbaseptr = RECNO()
   IF recnum <> mbaseptr
```

```
   recnum = mbaseptr
   @  row, col GET recnum;
               PICTURE "9999"
   CLEAR GETS
ENDIF
*
* ---We have source data so add a blank record to the destination file.
SELECT 3
APPEND BLANK
*
* ---Initialize source structure position.
SELECT 1
GO TOP
*
* ---If SRSXS then var1 = current block, var2 = current source structure record.
* ---If SRSXS then var1 = current source structure record, var2 = current block.
* ---dest = destination database field counter.
*
STORE 0 TO var1, var2, dest
*
* ---Loop through the source structure as long as there are fields
* ---required by the mode specified.
* ---If SRSXS then local1 = block count, local2 = source record count.
* ---If SXSSR them local1 = source record count, local2 = block count.
*
DO WHILE (var1 < local1) .AND. .NOT. EOF()
   var1 = var1 + 1
   var2 = 1
   DO WHILE (var2 <= local2)
      *
      * ---Select the source structure, increment the destination field
      * ---counter, store the current field name to a memory variable, and
      * ---compose the destination field name.
      *
      SELECT 1
      dest        = dest + 1
      msrc_field  = Field_name
      mfield_name = SUBSTR("B" + LTRIM(STR(var1)) + ;
                           "S" + LTRIM(STR(var2)) + ;
                           "D" + LTRIM(STR(dest)) + ;
                          TRIM(Field_name), 1, 10)
      *
      * ---Select the source database file and reset the base pointer.
      SELECT 2
      GO mbaseptr
      *
      * ---Compute offset to next block.  If the offset is not out of
      * ---range, jump to the offset for the current field, and transfer
      * ---the necessary data to a memory variable.
      *
      moffset = mbaseptr + mpage_len * (var&str1 - 1)
      IF moffset <= numrecs
         GO moffset
         &mfield_name = &msrc_field
         IF mmode = srsxs
            * ---Position to the next field name in the source structure.
              SELECT 1
            SKIP
         ENDIF
         * ---Select destination database file and fill in the data.
```

```
            SELECT 3
         REPLACE &mfield_name WITH M->&mfield_name
         ENDIF
         var2 = var2 + 1
      ENDDO
      *
      * ---Select source structure.
      SELECT 1
      IF mmode = srsxs
         GO TOP
      ELSE
         SKIP
      ENDIF
   ENDDO
   *
   * ---Select the source database file.
   SELECT 2
   *
   * ---Bump the line counter.  If it's time for a new page, reset line
   * ---counter, advance base pointer one data page, and initialize new
   * ---base page pointer.
   *
   mlincnt = mlincnt + 1
   IF mlincnt > mpage_len
      mlincnt = 1
      newptr = mbaseptr + mpage_len * (local&str1 - 1) + 1
      IF newptr < numrecs
         mbaseptr = newptr
         GO mbaseptr
      ELSE
         EXIT
      ENDIF
   ELSE
      *
      * ---Restore base pointer.
      GO mbaseptr
      SKIP
   ENDIF
ENDDO
SET ECHO OFF
*
* ---The output file is displayed here, using BROWSE.  If you do not wish
* ---to use this, just remove this section of the code.
SELECT 3
GO TOP
SET HELP OFF
BROWSE
*
RETURN
* ---EOP Snakgen.PRG
```

# Taking the Humbug Out of PROTECT



By Richard Bunter and James Bradbury

Databases are valuable commodities. They usually represent hours of toil and, more importantly, contain data that might be anything from irreplaceable to proprietary. That's why it makes sense to limit access to your data to those who genuinely need it. If you are using the dBASE Administrator, you have at your disposal an optional security utility—PROTECT—that is designed to do exactly that.

PROTECT provides three types of control over access to your database system:

- Log-in security
- File and field access security
- Data encryption

Unfortunately, many dBASE users find the prospect of setting up PROTECT a daunting one. To help such users get started with PROTECT, this article will examine a simple office scenario where a PROTECT system is being set up.

## Scrooge and Marley, Inc

Scrooge and Marley, Inc, has been using dBASE III PLUS since the product was introduced. The company's current CEO, Ebenezer Scrooge, considers himself a computer buff, having purchased one of the first inexpensive IBM-compatibles with money that he made by evicting a widow and her two crippled children.

Scrooge has decided to implement PROTECT to keep the company's highly sensitive files safe from prying eyes. Furthermore, he'd like to ensure that records aren't accidentally deleted by his shiftless clerk, Bob Cratchit. Scrooge didn't go back to school and earn his MBA for nothing — he first decides to sit down and figure out who in the company needs to be able to do what.

## Levels of PROTECT

Scrooge grudgingly decides that Bob Cratchit, who admittedly does almost all the work, will need full access to most fields, though there is one field that Scrooge doesn't want him to see at all. Furthermore, Cratchit definitely should not be allowed to delete any records.

The other major principal in Scrooge and Marley, Inc, is Scrooge's partner, Marley. Scrooge wants Marley's rights to be somewhat restricted, mainly because he's dead. In the end, Scrooge decides to give Marley full access to most fields, and read-only rights to the one field that Bob Cratchit can't even read. Like Cratchit, though, Marley is to have no delete privileges.

Finally, Scrooge reckons that he will need full access to all records and that he alone will be allowed to delete records. Now that he has a plan, he's ready to implement it.

Scrooge has already installed the dBASE Administrator on his hard disk; so, his next step is to define the network administrator's password. To do this, he types

```
PROTECT
```

at the DOS prompt.

He is then twice prompted to type the Network Administrator password, which can be any user-defined character string of up to 16 alphanumeric characters. Scrooge chooses "BAHHUMBUG".

Once the password has been successfully entered, Scrooge is presented with the User Menu, which is shown in Figure 1.

The "Account Name" field is optional, which means that dBASE doesn't need it at all. Its sole purpose is to allow the Administrator to add the full name of or a more descriptive title for the user (as the "Login Name" can't be more than 8 alphanumeric characters long). Scrooge tries unsuccessfully to remember Bob Cratchit's first name. After a moment, he types just "Cratchit."

The dBASE Administrator allows 8 different levels of access, with level 1 being the least restricted and level 8 being the most restricted. Of all the network users, Bob Cratchit should have the most restricted access rights, so Scrooge sets "Access level" to 8. Here's a recap of the values that Scrooge has typed in thus far:

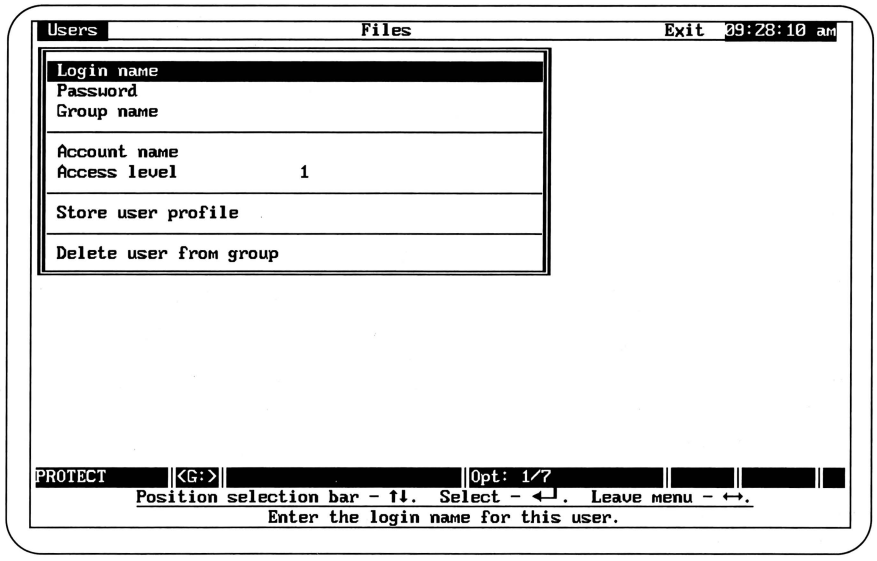



Figure 1

The next step is to list the users of the network (in 8 alphanumeric characters or less) and give them passwords. Scrooge decides to start from the bottom and work his way up.

Starting at the bottom means that he must begin with Bob Cratchit. Scrooge doesn't like to get too personal at the office, so he begins by typing "CLERK" in the field called "Login name."

PROTECT requires a password; so, Scrooge spends a few moments trying to think of one for Bob Cratchit. He settles on "OVERPAID" and types that into the field called "Password." For the field called "Group Name," Scrooge chooses "COMPANY", which is well within the limit of 8 alphanumeric characters.

```
Login Name: CLERK
Password: OVERPAID
Group name: COMPANY
Account Name: Cratchit
Access level: 8
```

With characteristic farsightedness, Scrooge realizes that it might be a good idea to have a written record of all the values that he has assigned, particularly as PROTECT does not have a reporting feature. At the end of Chapter 3, "Network Security" in the Network Administration section of the *Programming with dBASE III PLUS* documentation, Scrooge finds something called a "Security Request" form. He has Bob Cratchit make several copies of this and fills them out as he goes along.

## Rank and Privileges

Whether your security system is as straightforward as Scrooge's or not, the following tips should help you sort out the thorny questions that arise whenever rank and privileges must be assigned:

- PROTECT allows for access levels on users as well as files. If you have more than one file to PROTECT, however, it's wise to begin by thinking in terms of files, not users, because it's the way you set up the access levels within the files that will determine what kinds of rights each type of user will have.
- Encrypting a file involves three components: a group name, file privilege levels, and field privileges for each level. A user in a group other than the one that the file was encrypted under will have no access to the file at all.
- File privileges are actually a level at which a privilege becomes available. Since the most restrictive level is 8, if Delete privileges are set at 7, everyone who has an access level of 7 or less will be able to delete records, but the user at level 8 will not.
- Field privileges must be set for each level. If you don't set them for a given level, they will remain at the default setting (Full). If you encrypt the file without addressing the field privileges at each level, you could have a level 5 user who doesn't have the right to view certain fields that a level 6 user does. This could happen because you set the field privileges at level 5, but forgot about level 6.
- Many files can be encrypted under a single group name. Since the file privilege levels are set arbitrarily, you don't have to be consistent about how you set the levels in different files, although, for the sake of your sanity, you should be.
- The issue of access levels for users becomes somewhat murkier if you're dealing with multiple files. If one user must have very restricted access to one file but unrestricted use of another within the same group, what access level should you assign? Now add another user within the same group but with access needs that are the reverse of the first user. These are the sorts of issues that are best resolved *before* you start setting up your PROTECT system.

After selecting "Store User Profile," Scrooge proceeds to set up Marley. He assigns him a password: "SPECTRAL". His group name, too, is "COMPANY".

Scrooge wants Marley to have all the rights that Bob Cratchit does. In addition, though, he wants Marley to be able to access a field that Scrooge considers extra confidential and one that would not normally be accessible to Bob Cratchit. This means Marley will be midway between Bob Cratchit and himself; therefore, Scrooge assigns him an Access level of 5. Here are the final values:

```
Login Name: MARLEY
Password: SPECTRAL
Group name: COMPANY
Account Name: Marley's Ghost
Access level: 5
```

Finally, Scrooge comes to himself. He needs full access to everything. He sets the following values:

```
Login Name: SCROOGE
Password: THRIFT
Group name: COMPANY
Account Name: Ebenezer Scrooge
Access level: 1
```

Now that he has defined all three types of user, Scrooge moves to the **Exit** menu and saves the **Users** information. The next step is to define how he wants the system to treat his particular file—Debtors.DBF.

## Select the File

After restarting PROTECT, Scrooge selects **Files** from the PROTECT menu and sees the screen shown in Figure 2.

The "Select new file" option is where Scrooge chooses which database file he wants to encrypt. When he presses **Return**, he sees a window in the upper-right corner of his screen that tells him what files are available on the currently selected drive. He chooses Debtors.DBF. This means that the system will make an encrypted copy of that file with the name Debtors.CRP.

## File Access Privileges

The next step is to establish the file privilege levels. At this stage, Scrooge will determine two things:

- Who has access to what records
- What they can do to those records

Under "File access privileges" are listed five parameters:

- Group name
- Read privilege level
- Update privilege level
- Extend privilege level
- Delete privilege level

The group name specifies what set of users these privileges will apply to. As this is a simple example, there is only one group: COMPANY.

The privilege level parameters are threshold values. This means that an ability becomes available to users who are at a certain level and is automatically available to all the users who are above that level. For instance, if level 5 is defined as being able to DELETE records, then levels 5, 4, 3, 2, and 1 will all be able to DELETE records. For more on this concept, see the accompanying sidebar, "Rank and Privileges."

At Scrooge and Marley, Inc, the Debtors database file has to be accessible to all the users, so even the most restrictive level must be able to view all the records. Therefore the Read privilege is set to 8, which means that everyone with an Access level of 8 or higher (which paradoxically means a lower number) can read records in the Debtors database file.

Because everyone should also be able to APPEND and EDIT records, Extend and Update privileges are both set to 8 also. At this point, no one has been denied access to any record. If he had set the Read privilege at a higher level than 8, then Bob Cratchit wouldn't have been able to access those records at all. As it is, he has given Cratchit the right not only to read records, but to change them.

However, Scrooge wants to restrict the right to DELETE records to himself. For this reason, he sets the Delete privilege to 1. It's worth noting that the only way a file privilege can be restricted to just one set of users is to have that set of users at the very top of the hierarchy.

## Field Access Privileges

Under "Field access privileges" are two fields:

- Establish access level
- Establish field privileges

This is where the Network Administrator, Scrooge, determines what different sets of users can do to specific *fields* of the database records. This is separate from what they can do to the *records* of the database file; that was determined under "File access privileges."

The Debtors database file has one field that Scrooge considers sensitive: BALANCE. He's worried that if Bob Cratchit knows how much the company is making from evictions, he'll want a raise. He doesn't want Cratchit to be able to access that field. Although he doesn't mind if Marley accesses it, Scrooge doesn't want Marley to be able to change it. You can trust a ghost only so far. Scrooge doesn't want anyone other than himself to meddle with BALANCE.

Scrooge cursors to "Establish access level" and presses **Return**. He then enters 8 because he is establishing field access privileges for Cratchit. After cursoring to "Establish field privileges" and pressing **Return** again, he is presented with a window that lists all the fields of the database file and their access levels. The default for all the fields is

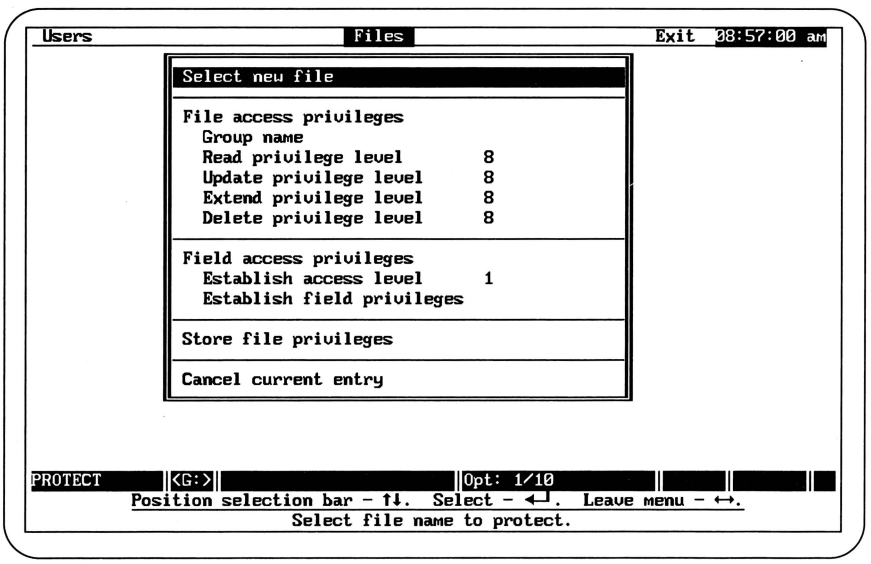



Figure 2

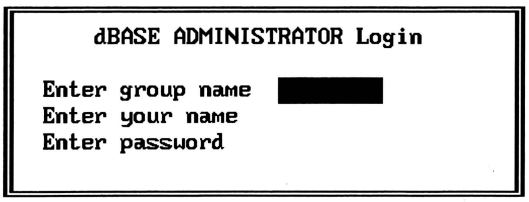



Figure 3

FULL, with the other two options being NONE and R/O (read-only).

For level 8, the field privileges should all remain defined as FULL except for BALANCE, which should be NONE. Once he's satisfied with the entries, Scrooge cursors back to the main **Files** menu and selects "Store file privileges." This process is then repeated for the other two levels of users.

At level 5, Marley's level, all fields should be FULL except for BALANCE, which should be R/O (Read Only). This means that Marley will be able to read but not change the data in that field.

At level 1, Scrooge's level, all the fields should be FULL.

Once everything has been defined, Scrooge exits from the PROTECT screen by choosing **Exit:Save**.

## Invoking Protect

All Scrooge's efforts are stored in a file called DBSYSTEM.DB. If that file does not exist in the current directory when PROTECT is invoked, it is created. Otherwise, DBSYSTEM.DB will be backed up under the name $ (a single dollar sign with no extension).

It is the presence of DBSYSTEM.DB in the directory that tells the Administrator that PROTECT is in effect. Otherwise users would not be prompted to log in, and the files would not be automatically unencrypted once a login was successful. For this reason, too, you must access the Administrator from the directory that contains DBSYSTEM.DB. If DBSYSTEM.DB is *not* in your current directory, you will enter dBASE but your files will not be unencrypted.

To enhance the security of his newly encrypted database, Scrooge does the following:

1. He makes a safety copy of the Debtors.DBF file and stores it in a safe place.
2. He copies the .CRP file to the .DBF file with the following DOS command:

```
COPY Debtors.CRP Debtors.DBF
```

The unencrypted .DBF file must be deleted from the disk because otherwise dBASE would open that file instead of the encrypted one, which would defeat the security system rather neatly. The reason for COPYing the .CRP file over the .DBF file instead of just erasing the original .DBF file is to prevent some unscrupulous person from unerasing the unencrypted .DBF file. Even Scrooge has heard of the Norton Utilities.

Now, when any employee of Scrooge and Marley, Inc, accesses the dBASE Administrator (by typing "DBA", he or she will be presented with the screen shown in Figure 3.

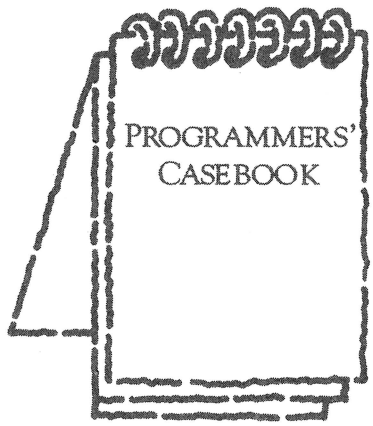
When the user has typed in the three required parameters, the system will be able to establish what Access level that user has. The user will then have access to the database, but only in the manner prescribed by the Network Administrator.

## Adding Users

When the Network Administrator wants to add users to the security system, he follows these steps:

1. Access the User Menu.
2. Enter a user login name.
3. Enter a user password.
4. Enter a group name.
5. Enter the account name (optional).
6. Select a user access level.
7. Store the user profile.

# Try RETRY Again




By Kenneth N. Getz

In the February 1987 issue of ***TechNotes***, we published the following dBASE III PLUS anomaly :

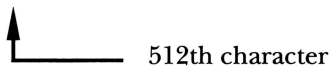
---

**RETRY**

When a program contains ON ERROR RETRY or ON ERROR DO <program> and <program> contains RETRY, the command line executed by RETRY may be truncated when RETRYed.

If the line that causes the error spans two 512-byte blocks counting from the beginning of the command file, the line is truncated at the 512th character when RETRY is executed. For example,

STORE REPLICATE('A',<n>) TO mem




When RETRY is executed, dBASE III PLUS tries to execute the line as

'A',<n> TO mem

dBASE III PLUS executes only that part of the command in the latter 512-byte block. This in turn causes another error, which puts the program in an endless loop.

---

When that anomaly was listed, no consistent workaround to correct the situation was known. The best method was to go through your file and make sure that the 512th byte of every block fell on a comment, a carriage return, or a linefeed. While this would solve the problem, it was far too tedious to be a practical workaround.

The program presented here, Retry.C, will *automatically* ensure that every 512-byte block in your program code ends in a safe spot. You should run this program on all your program code once your code is in its final, debugged state. If you make any changes to your program, no matter how minor, you will have to run Retry again, as adding a single byte to your dBASE program could undo Retry.C's efforts on your behalf.

## How Retry Works

Retry performs, under program control, the same steps you would follow if you were going to check and prepare your code yourself. Here is the pseudocode for what the program does:

```
For each file to be processed,
      While a block can be read,
            If the block is less than 512 bytes or ends with a CR/LF,
                  just write the block to the output file.
            Else
                  Walk backward in the block, looking for a LF.
                  If we find one within 2 characters, walk back and find another.
                  Assume we're on a continuation line.
                  While the current line ends with a ';'
                        Starting from current position, look backward for a ';'.
                        Walk back in the block looking for a LF.
                  Now that we've found a LF that won't break a continuation line,
                        Write the part before the LF out to the output file.
                        Write out enough asterisks, then a CR/LF, to force the
                              CR/LF to fall on the 511th and 512th bytes.
                  Back up in the input file to account for the asterisks added.
```

There are, of course, other methods you could follow to accomplish the same goal, but this most closely represents how I would do it by hand.

## Setup

To create Retry.EXE, first type in the program code found in this article using an ASCII text editor (or the Turbo C integrated environment). The comments (surrounded by /* and */) are included for clarity only, and so you don't have to type them if you don't want to. The program has been tested and compiled with both Microsoft C, version 4.0 and Borland International's Turbo C, version 1.0. If you're using Turbo C, you should type in both Retry.C and Wsetargv.C. Microsoft C users will need to type in only Retry.C. The program Wsetargv.C has been generously contributed by Michael Rubenstein of Reid Associates, Arlington, VA. It allows Turbo C users to pass wildcard parameters on the command line. (Microsoft C users will find the program to do this included with the compiler, in the file xSETARGV.C, where the "x" stands for the memory model. This should be "S" for this program.)

Before you use either compiler to produce executable code, you must have installed it according to the manufacturer's instructions. Please refer to your compiler's installation instructions.

Once the source code has been typed in, the program can be compiled and linked with the following instructions:

**Microsoft C:**

```
msc retry;
link retry+ssetargv;
```

**Turbo C:**

```
tcc retry wsetargv
```

In either case, you will end up with Retry.EXE, which can be run from the DOS prompt.

## Using Retry

To use Retry, place it either in the directory with the programs to fix, or in your DOS path. At the DOS prompt, type

```
RETRY filespec
```

where "filespec" can be any legal DOS filename, including path names and wildcards. For example, to fix all the .PRG files in your current directory, you could type

```
Retry *.PRG
```

## Retry.C

```
/*************************************************************************/
/*  File.....: Retry.C                                                   */
/*  Author...: Kenneth N. Getz                                           */
/*  Date.....: December 1, 1987                                          */
/*  Notes....: Retry.C attempts to work around the 512-byte RETRY anomaly   */
/*             in dBASE III PLUS.  It works by making sure each 512-byte    */
/*             section of the input file is padded with enough asterisks in */
/*             the output file to force each new 512-byte section to end    */
/*             with a CR/LF.                                                */
/*************************************************************************/

#include <stdio.h>
#include <string.h>
#include <process.h>

int  openfiles(char *, char *);
void process(void);
int  fix(char *);
void makechange(char *);

#define FALSE 0
#define TRUE !FALSE
#define SIZE 512
#define SP 32
#define LF 10
#define CR 13
#define TAB 9
#define WALKBACK  while (buffer[SIZE - fillbytes - 1] != LF)  \
                     fillbytes++


char buffer[SIZE];
FILE *infile;
FILE *outfile;

main(argc, argv)
int argc;
char *argv[];
{
   char newfile[65];
   char filename[65];

   if (argc == 1)              /* If there were no command-line parameters, */
      {
      printf("Filename to process (wildcards not allowed): ");
      gets(filename);                    /* Get the filename to operate on. */
      if (strlen(filename) == 0)            /* If user just pressed Return, */
         exit(0);                                          /* Quit now. */
      if (openfiles(filename, newfile))    /* If files opened successfully, */
         makechange(newfile);                            /* Do the work. */
      }
   else while (*(++argv))                /* If files given on command line, */
      if (openfiles(*argv, newfile))                       /* open files, */
         makechange(newfile);                        /* and do the work. */
   return(0);
}
```

```
void makechange(newfile)
char *newfile;
{
   process();
   printf("%s\n", newfile);
   fcloseall();
}

int openfiles(filename, newfile)
char *filename;
char *newfile;
{
;
/* Attempt to open the input and the output files.  The output filename is  */
/* created by changing the last character of the input filename to a '!'.   */
/* For the sake of simplicity, this does no checking to make sure that the  */
/* output file doesn't already exist, but you could easily add that check.  */

   strcpy(newfile, filename);          /* Make a copy of the input filename. */
   newfile[strlen(newfile)-1] = '!';          /* Change the last character. */
   printf("%-15s --> ", filename);                      /* Print a prompt. */
   if ((infile = fopen(filename,"rb")) == NULL)     /* If we can't open it, */
      {
      printf("Not found.\n");                          /* indicate an error. */
      return(0);                                               /* And quit. */
      }
   if ((outfile = fopen(newfile,"wb")) == NULL)    /* If can't open output, */
      {
      printf("Cannot open output file.\n");            /* indicate an error. */
      return(0);                                               /* And quit. */
      }
   return(1);
}

void process()
{

/* Process the file in 512-byte chunks.  This (handily) corresponds to the  */
/* size of dBASE III PLUS's program buffer.  We continue reading until we   */
/* can't read anything else from the file.  If, at any point, we get a      */
/* chunk less than 512 bytes, we assume that we're at the end and write     */
/* that chunk out to the output file directly, with no processing.          */

   int bytesread;

   while((bytesread = fread(buffer, 1, SIZE, infile)) != 0)
      {
      if ((bytesread < SIZE) || (buffer[SIZE-2] == CR && buffer[SIZE-1] == LF))
         fwrite(buffer, 1, bytesread, outfile);
      else fix(buffer);
      }
}

int fix(buffer)
char *buffer;
{
```

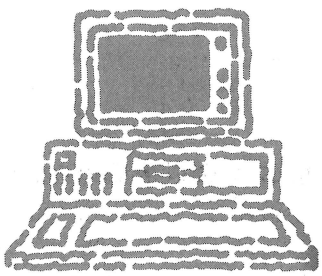
```
/* Walk backward through buffer looking for last CR/LF.  Once it's found,   */
/* the program must check to make sure it isn't breaking a continuation     */
/* line, indicated with a ";" at the end of the previous line.  If the      */
/* program finds a ";", it repeats the process, looking for a CR/LF until   */
/* it finds one that won't break a continuation.  We're guaranteed to find  */
/* a safe break somewhere in the 512-byte buffer, since the command line is */
/* limited to 254 characters.                                               */

   int fillbytes = 0;
   int i, sameline, q, ch;

   WALKBACK;                        /* Walk backward, looking for linefeed.   */
   if (fillbytes < 2)             /* If we didn't get at least two characters, */
      {
      fillbytes += 2;               /* Repeat the process back another line. */
      WALKBACK;                        /* Walk back and look for another LF. */
      }
   sameline = TRUE;                  /* Assume we're on a continuation line. */
   while (sameline)                   /* While we're on a continuation line, */
      {
      q = 0;                                                  /* Reset counter. */
      while (((ch = buffer[SIZE - fillbytes - q - 3]) == SP) ||
              (ch == TAB))              /* Skip over ending spaces and tabs. */
         q++;
      sameline = (buffer[SIZE - fillbytes - q - 3] == ';');
      if (sameline)                              /* If we found a semicolon,*/
         {
         fillbytes += (q + 2);                          /* Account for CR/LF. */
         WALKBACK;                        /* Walk backward, looking for LF.  */
         }
      }
   /* Now we write out the first part of the buffer, out to the end of the  */
   /* last full, non-continuation line.                                     */
   fwrite(buffer, 1, SIZE - fillbytes, outfile);
   for (i = 0; i < fillbytes - 2; i++)        /* Fill with enough asterisks. */
      fputc('*', outfile);
   fputc(CR,outfile);                     /* Put the trailing CR and LF out. */
   fputc(LF,outfile);
   fseek(infile, (long)(-fillbytes), SEEK_CUR);         /* Back up enough to */
   return(0);                           /* make up for the asterisks we added. */
}
```

## Wsetargv.C

```
/*************************************************************************/
/*  File.....: Wsetargv.C                                                */
/*  Author...: Michael M. Rubenstein                                     */
/*  Notice...: Copyright 1987, All rights reserved.                      */
/*************************************************************************/
/* This file implements wild card expansion in argv for Turbo C.         */
/* Strings of characters in either quotes (") or apostrophes (') on the  */
/* command line are considered a single argument.  However, backslash    */
/* escapes are not implemented.  A quote may be included in an argument  */
/* which is enclosed in apostrophes and an apostrophe may be included    */
/* in an argument enclosed in quotes.  Either may be included in an      */
/* argument starting with any other character.                           */
/*                                                                       */
/* Any argument which is not enclosed in quotes or apostrophes, does not */
/* begin with a hyphen (-), and which contains an asterisk (*) or question */
/* mark (?) will be expanded.  It is NOT an error for an argument to have a */
/* null expansion (no matching files).  Only ordinary files (not         */
/* directories or hidden or system files) will be included in the        */
/* expansion.                                                            */
/*                                                                       */
/* To use this function, simply compile it with the appropriate memory   */
/* model and include in the link.  This can be accomplished very simply  */
/* in the integrated environment by simply including this file in the    */
/* project file.  In the command line version, simply include this file  */
/* (or a precompiled .OBJ version) on the command line.                  */
/*************************************************************************/

#include <ctype.h>
#include <dir.h>
#include <dos.h>
#include <process.h>

#define FALSE           0
#define TRUE            1

void putarg(unsigned char far *, unsigned char far *);

extern int           __argc;
extern char          far **__argv;
extern unsigned      _psp;
extern unsigned      _envseg;
extern unsigned      _envLng;
extern unsigned char _osmajor;
extern void          _abort();
extern char          *sbrk(int);

void _setargv()
{
   unsigned char far *cmdtail;
   unsigned char     *firstarg;
   unsigned char far *cmdarg;
   unsigned char     buffer[129];
   unsigned char     *p, *q;
   unsigned char     *lastdir;
   char **wargv;
   int wild;
   int c;
   int i;
   struct ffblk ffb;
```

```
   cmdtail = MK_FP(_psp, 0x81);
   cmdtail[cmdtail[-1]] = '\0';     /* Make sure there's a null at the end. */
   firstarg = sbrk(0);
   __argc = 1;

   while (*cmdtail != '\0')
      {
      while (isascii(*cmdtail) && isspace(*cmdtail))   /* Skip white space. */
         ++cmdtail;
      if (*cmdtail == '\0')      /* If end of command tail, done with loop. */
         break;

                               /* If quoted string, just save the argument. */
      if ((c = *cmdtail) == '"' || c == '\'')
         {
         cmdarg = ++cmdtail;
         while (*cmdtail != c && *cmdtail != '\0')
            ++cmdtail;
         putarg(cmdarg, cmdtail);
         if (*cmdtail != '\0')
            ++cmdtail;
         continue;
         }
      cmdarg = cmdtail;                                    /* Find word. */
      wild = FALSE;
      p = lastdir = buffer;
      while ((c = *cmdtail) != '\0' && (!isascii(c) || !isspace(c)))
         {
                                  /* Wild is TRUE if word contains * or ? */
         wild |= (c == '*' || c == '?');
         *(p++) = c;
            /* Lastdir points to the first character of the base filename.  */
         if (c == '\\' || c == ':')
            lastdir = p;
         ++cmdtail;
         }
      *p = '\0';
      if (wild && *cmdarg != '-')
         for (c = findfirst(buffer, &ffb, 0); c == 0; c = findnext(&ffb))
            {
             /* Use lowercase for wild card expanded names (my prejudice).  */
            for (p = lastdir, q = (unsigned char *) ffb.ff_name; *q != '\0';)
               *(p++) = tolower(*(q++));
            putarg(buffer, p);
            }
      else putarg(cmdarg, cmdtail);
   }

   /* Allocate argv. */
   if ((wargv = (char **) sbrk(sizeof(char *) * (__argc + 1))) == (char **) -1)
      abort();
   __argv = wargv;
   if (_osmajor < 3)                                 /* Store program name. */
      *(wargv++) = "C";
   else
      {
      cmdtail = cmdarg = MK_FP(_envseg, _envLng + 2);
#     if defined(__TINY__) || defined(__SMALL__) || defined(__MEDIUM__)
         *(wargv++) = sbrk(0);
```

```
            while (*cmdtail != '\0')
               ++cmdtail;
            putarg(cmdarg, cmdtail);
            --__argc;
#        else
            *(wargv++) = cmdarg;
#        endif
         }
      for (i = __argc; --i;)                          /* Store arguments. */
         {
         *(wargv++) = firstarg;
         while(*++firstarg != '\0')
            ;
         ++firstarg;
         }
      *wargv = (char *) 0;
}

static void putarg(from, to)
unsigned char far *from, far *to;
{
   char *p;

   if ((p = sbrk(to - from + 1)) == (char *) -1)
      abort();
   while (from < to)
      *(p++) = *(from++);
   *p = '\0';
   ++__argc;
}
```



# dBASE TIPS

By Chuck Litzell

## Manipulating Dates for Business

Some financial and accounting calculations need to consider the number of working days in a period as opposed to the number of calendar days. In the November 1986 ***TechNotes***, Weekday2.PRG, a program to calculate the number of work days between two dates, was presented.

Weekday2 returns an incorrect number of days if the ending date occurs on a day of the week earlier than the beginning day. For instance, if the ending day is a Monday and the beginning day is a Wednesday, the answer will be incorrect. Weekdays.PRG (p. 36) is a corrected version of Weekday2.

To set up Weekdays.PRG, first CREATE Holiday.DBF with the following structure.

```
Structure for database:  Holiday.DBF
   Field  Field Name   Type    Width   Dec
     1    Holiday      Date       8
```

Then, APPEND all the holidays that span the range of dates in which you will be working. This means that if you are working with a span of two years, you must have APPENDed two records for each holiday, one for each specific date.

To execute Weekdays.PRG (page, 36), use the following general syntax after initializing a numeric return variable for the number of work days returned from the routine.

```
ret_val = 0
DO Weekdays WITH <start date>, <end date>,
ret_val
```

## More Date Calculations

Sometimes a program needs to know the last day of a month. The following code fragment adjusts a date memory variable to the last day of the month.

```
* ---Make mdate the last day of the month.
mdate = CTOD(STR(MONTH(mdate) + 1, 2) +;
       "/01/" + STR(YEAR(mdate), 4)) - 1
```

This code works by constructing the date for the first day of the following month and then subtracting one day from that date. It is interesting to note that to calculate the last day of December 1987 the formula would evaluate the expression

```
CTOD("13/01/88") - 1
```

dBASE III PLUS correctly converts dates like this one to valid dates.

Another important business day is the first Monday in a month. The code below adjusts a date memory variable to the first Monday in the month.

```
* ---Make mdate the first Monday of the
month.
mdate = mdate - DAY(mdate) + 1
mdate = mdate + ;
   VAL(SUBSTR("1065432", DOW(mdate), 1))
```

The first line changes the date to the first day of the month. The second line adds the number of days required to move the date up to Monday. It uses the DOW() function and a table stored in a character string to find the number of days that need to be added to move the date to the first Monday. For example, if the first of the month falls on a Friday, DOW() returns 6. The SUBSTR() function returns the sixth character of "1065432", which is "3". Monday is 3 days from Friday. Adding the VALue of "3" to mdate yields the date of the following Monday.

## Indexing and Signed Numbers

If you try to index on a character field and the STR() of a field that contains both positive and negative numbers, the negative and positive values will be mixed (Figure 1).

The trick is to "scale" all the numbers in the index key so that they are positive, by adding a constant larger than the smallest negative number. This doesn't change the database file, but it does cause

```
USE NUMBERS                          INDEX ON Letter+STR(Number,5,0) TO Numbers
LIST                                 LIST
Record#  LETTER   NUMBER             Record#  LETTER   NUMBER
      1  A         -3456                   3  A           34
      2  A          -345                   4  A          345
      3  A            34                   2  A         -345
      4  A           345                   5  A         3456
      5  A          3456                   1  A        -3456
```

Figure 1

## Weekdays.PRG

```
* Program ...: Weekdays.PRG
* Author ....: Kenneth N. Getz
* Revised by : Steven Kurasch
* Date ......: December 1, 1987
* Version ...: dBASE III PLUS and Developer's Release
* Note(s) ...: Program to find number of working days given a
*               starting and ending date. Includes both enddates.
*
*               DO Weekday2 WITH begdate, enddate, weekdays
*
*                  begdate  := starting date
*                  enddate  := ending date
*                  weekdays := numeric memvar to receive answer
*
PARAMETERS begdate, enddate, weekdays

* ---If begdate falls on a weekend, push it up to Monday.
begdate = begdate + VAL(SUBSTR("1000002", DOW(begdate), 1))

* ---If enddate falls on a weekend, back it up to Friday.
enddate = enddate - VAL(SUBSTR("2000001", DOW(enddate), 1))

* ---Days between dates.
days = enddate - begdate + 1

* ---Subtract out weekends.
weekdays = days - INT(days / 7) * 2

* ---Adjust for partial week.
IF DOW(enddate) < DOW(begdate)
   weekdays = weekdays - 2
ENDIF

* ---Use Holiday.DBF to count number of weekday holidays
* ---between dates.
USE Holiday
COUNT to counter FOR (Holiday >= begdate;
                 .AND. Holiday <= enddate);
                 .AND. MOD(DOW(Holiday) - 1, 6) <> 0
USE
weekdays = weekdays - counter
RETURN
* EOP Weekdays.PRG
```

the index file to contain different information than you would expect (Figure 2).

To do a SEEK or FIND becomes a bit more complicated. You must remember to scale the number in the SEEK or FIND expression as well.

```
. SEEK "A" + STR(345 + 9999, 5, 0)
```

## Blank Lines in a Label Format

The label generator in dBASE III PLUS condenses labels by removing blank lines. You may want a label format with blank lines. For instance, you could print a customer's account number two lines beneath the address.

The trick is to put something in the Label Contents that will not print and is not a space. CHR(255), often interpreted as a hard space character, prints as a space on an IBM PC display and on most printers. It's a good choice if labels are intended for screen output.

When you're concerned only with printing labels on the printer,

```
" " + CHR(8)
```

will provide satisfactory results on the widest variety of printers. CHR(8) is the backspace character. This expression will cause the printer to print a space, then backspace over it.

## COPY TO

Trying to COPY a database file with a memo field to another file with the same name, but a different extension, returns the error message "File is already open." This happens because the new memo file has the same name as the old memo file.

## CREATE REPORT

The maximum column width that REPORT FORM defaults to is 30 characters. If a longer expression is being entered, the column width must be changed manually.

## dCONVERT

dCONVERT version 2.01 does not properly convert command lines that are missing a period in double Boolean operators. For example.

```
STORE T TO mflag
DO WHILE mflag .AND .NOT. EOF
                 ↑
                 └── Missing period
```

Because the required character is missing, "EOF" is not converted to "EOF()."

```
INDEX ON Letter + STR(Number + 9999,5,0) TO Numbers
LIST
Record#  LETTER   NUMBER
      1  A         -3456
      2  A          -345
      3  A            34
      4  A           345
      5  A          3456
```

Figure 2

## ERROR()

With an ON ERROR in effect (Figure 3), ERROR() returns –1 for the error messages "Not enough records to SORT." and "No database is in USE. Enter file name:". None are trapped by ON ERROR.

## FILE()

In DBASE III versions 1.0 and 1.1, the FILE() function allowed the use of wild card characters. In the Developer's Release and dBASE III PLUS, the FILE() function always returns a false when wild card characters are used. For example,

```
* DBASE III
? FILE('DBASE.*')
.T.

* DBASE III PLUS and Developer's Release
? FILE('DBASE.*')
.F.
```

## Logical Operator

dBASE III PLUS may improperly parse a command line when the line includes a misspelled logical operator. When a command line contains a misspelled operator, the error message "Unrecognized phrase/keyword in command."[36] should be produced when the operator is encountered. The evaluation of the command line varies depending on the command and the expression in which the illegal operator is included. For example,

```
? .T. .NAD. .F.
```

correctly returns "Unrecognized phrase/keyword in command" [36]. But, the following do not.

```
* ---Evaluates as true.
IF .T. .NAD. .F.
* ---"Unbalanced parenthesis." [8]
IF (.T. .NAD. .F.)
* ---"Syntax error." [10]
DO WHILE .T. .NAD. .F.
```

```
ON ERROR ? ERROR()
? ERROR()                                   && Returns 0.
LIST
No database is in USE.  Enter file name:
? ERROR()                                   && Displays -1 if no name entered.
                                            && Displays 0 if a name is entered.
* ---Reset ERROR() to a positive number.
DUCK                                        && Displays 16.
USE Test.DBF
ZAP                                         && Delete all records.
SORT ON Name TO Test1
Not enough records to sort.
? ERROR()                                   && Displays -1
```

Figure 3

## SET CATALOG ON

dBASE III PLUS does not prompt for a catalog title when a catalog file is deleted from the disk and a new catalog file with the same name is created. This happens because the catalog filename is not deleted from the master catalog. To remove catalogs from the master catalog, edit CATALOG.CAT and DELETE the appropriate record, then PACK or, issue SET CATALOG TO ?. When SET CATALOG TO ? is executed, catalog files that are no longer accessible are deleted from the master catalog.

## TRANSFORM()

Using the TRANSFORM() function in a REPORT FORM to format numeric output does not allow totaling of the contents of that report column. The output that TRANSFORM() generates is character data, and therefore cannot be totaled.

## USE...ALIAS

If a filename starts with a numeral, it cannot be used as an ALIAS, although DISPLAY STATUS shows it as the default alias name. An attempt to use such a filename as an ALIAS returns the "Unrecognized phrase/keyword in command" error message. For example,

```
USE 1test
SELECT 2
? 1test->Field
Unrecognized phrase/keyword in command.
```

## Converting to Initial Caps

By James Chuang

It is common practice to enter data into computers in all capital letters. It's faster and you don't need to worry about the state of the **CapsLock** toggle. One day, though, you'll want to transfer data from a dBASE III PLUS database file to a word processor to generate merged form letters. Then it would be nice to have the names and addresses with initial caps, as they would appear on a letter.

Icaps.PRG is a program that converts a string to initial caps. It extracts each space-delimited word from the string, uppercases the first letter, and lowercases the rest. Icaps looks for common places where a letter in the middle of a word should be capitalized.

If a word begins with "MC," the "C" is lowercase, and the letter following the "C" is capitalized, as in Old McDonald.

When an apostrophe occurs within the first three letters of a word, the letter following the apostrophe is capitalized, as in O'Rourke.

Letters following hyphens, periods, or ampersands occurring anywhere within a word are capitalized. Icaps is called with the syntax

```
DO Icaps WITH <memvar>
```

A short program that calls Icaps can convert an entire database to initial caps.

```
USE <database>
DO WHILE .NOT. EOF()
   temp = <field to convert>
   DO Icaps WITH temp
```

```
   REPLACE <field to convert> WITH temp
    SKIP
ENDDO
```

Icaps.PRG will take care of the bulk of the conversion. However, a manual inspection of the results will be required to deal with other exceptions.

## Retrieve the Index Key Expression

By J. Mike Hedblom

The program described here, Getkey.BIN, allows the dBASE III PLUS user to find the key expression of an index file. The key expression can always be determined in dBASE III PLUS by first opening the database that relates to the index file. However, if the matching database file is unknown or is not available, the index cannot be opened. This program, Getkey.BIN, will allow you to find the index key expression under any circumstances, as long as the index file is available.

### Setup

To create Getkey.BIN, use an ASCII text editor to enter the assembly language code to a file named Getkey.ASM. You should also use your text editor to create the generic batch file Assemble.BAT, which you can use to assemble any .BIN file.

To invoke Assemble.BAT, type

```
ASSEMBLE Getkey
```

at the DOS prompt. Masm.EXE, Link.EXE, and Exe2bin.EXE must all be available in the current directory or in your DOS path. You should type only the .ASM filename, not the extension.

### Usage

To use Getkey.BIN, CALL Getkey WITH a 100-character variable that contains the name of the index file. The variable must have the filename padded with spaces. For example,

```
mvar = LEFT('Zipcode.NDX' + SPACE(100), 100)
```

Note, the extension (.NDX) is required. DOS drive and directory specifications are optional.

```
mvar = LEFT("C:\DBASE\SUB\Filename.NDX" +;
                SPACE(100), 100)
```

To call Getkey, execute the commands

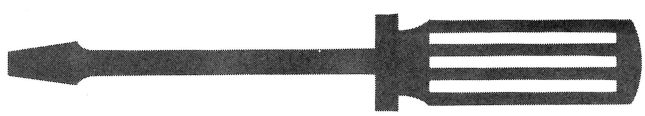
```
LOAD Getkey
CALL Getkey WITH mvar
```

The call to Getkey returns the index key expression in mvar. If an error occurs (such as 'File not found'), the first character of mvar is CHR(228). For example,

```
IF LEFT(mvar, 1) = CHR(228)
   ? "An error occurred"
ELSE
   ? mvar
ENDIF
```

### Assemble.BAT

```
REM Assemble.BAT
ECHO OFF
CLS
IF "ZIP%1" == "ZIP" GOTO ERROR3
MASM %1,%1,%1.LST;
IF ERRORLEVEL = 1 GOTO ERROR1
LINK %1,%1,NUL;
IF ERRORLEVEL = 1 GOTO ERROR2
ECHO It is proper to ignore
ECHO the previous error message.
EXE2BIN %1.EXE %1.BIN
GOTO EXIT
:ERROR1
ECHO ASSEMBLER ERROR!
GOTO EXIT
:ERROR2
ERASE %1.EXE
ECHO LINKER ERROR!
GOTO EXIT
:ERROR3
ECHO ASSEMBLE requires a filename.
ECHO Please try again.
ECHO Example: ASSEMBLE FOO
:EXIT
```



## Icaps.PRG

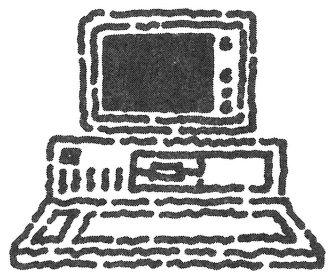
```
* Program ...: Icaps.PRG
* Author ....: James Chuang
* Date ......: December 1, 1987
* Versions ..: dBASE III PLUS, 1.0 and 1.1
* Notes .....: Uppercases the first letter of each word in a
*              memory variable.
*
*              Usage:  DO Icaps WITH <memvar>
*
PARAMETERS string
*
oldstr = LTRIM(TRIM(string)) + " "
string = ""
DO WHILE LEN(oldstr) > 1
   tempstr = UPPER(LEFT(oldstr, AT(" ", oldstr)))
   oldstr + LTRIM(STUFF(oldstr, 1, LEN(tempstr), ""))
   IF .NOT. ("&" $ tempstr .OR. ['] $ tempstr .OR.;
                  "." $ tempstr .OR. "-" $ tempstr)
      tempstr = UPPER(LEFT(tempstr, 1)) + LOWER(SUBSTR(tempstr, 2))
   ELSE
      counter = 1
      capnext = .T.
      tempstr2 = ""
      DO WHILE counter <= LEN(tempstr)
         pchar + SUBSTR(tempstr, counter, 1)
         DO CASE
         CASE capnext
            tempstr2 = tempstr2 + UPPER(pchar)
            capnext = .F.
         CASE AT(pchar, ".-") > 0
            capnext = .T.
            tempstr2 = tempstr2 + pchar
         CASE pchar = [']
            capnext = IIF((LEN(tempstr) - counter > 3), .T., .F.)
            tempstr2 = tempstr2 + "'"
         CASE pchar = "&"
            tempstr2 = UPPER(tempstr)
            EXIT
         OTHERWISE
            tempstr2 = tempstr2 + LOWER(pchar)
         ENDCASE
         counter = counter + 1
      ENDDO
      tempstr = tempstr2
   ENDIF
   IF AT("Mc", tempstr) = 1 .AND. LEN(TRIM(tempstr)) > 2
      tempstr = "Mc" + UPPER(SUBSTR(tempstr, 3, 1)) + SUBSTR(tempstr, 4)
   ENDIF
   string = string + tempstr
ENDDO
string = TRIM(string)
RETURN
* EOP Icaps.PRG
```

## Getkey.ASM

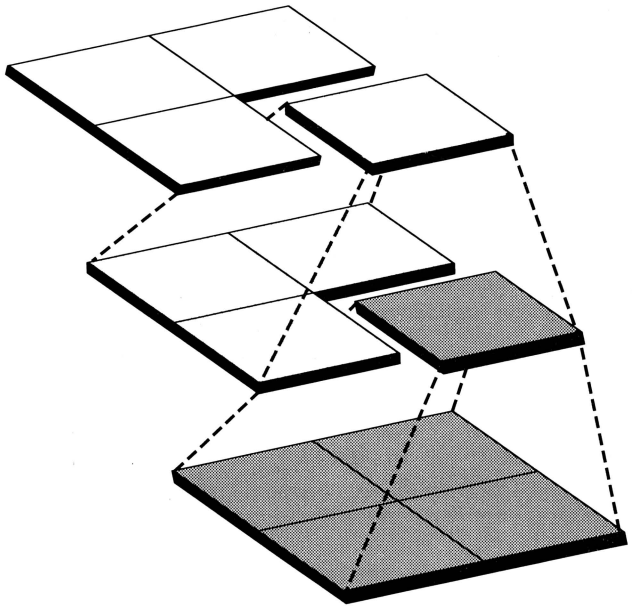
```
; Program ....: Getkey.ASM
; Author .....: J. Mike Hedblom
; Date .......: December 1, 1987
; Version(s) .: dBASE III PLUS and Developer's Release
; Note(s) ....: Takes the contents of a dBASE III PLUS .NDX Key
;               expression and returns it in a variable string.
;
;               If an error occurs, the variable is returned with
;               an ASCII E4h (CHR(228)) only.
;
;               The variable should be initialized with the index
;               filename left justified and padded with spaces to a
;               length of 100 characters.
;
;       USAGE:  mvar = LEFT("Zipcode.NDX"+SPACE(100),100)
;               CALL GETKEY WITH mvar
;               IF LEFT(mvar,1) = CHR(228)
;                 * Error occurred...
;               ENDIF
;
SIZE    EQU     125                     ; From start of .NDX to end of Key.
ERRCHAR EQU     0E4h                    ; Error character.
NULL    EQU     0                       ; ASCII null.
SPACE   EQU     32                      ; ASCII SPACE.
KEYLEN  EQU     100                     ; Length of index key expression.
CODE    SEGMENT BYTE    PUBLIC  'CODE'
GETKEY  PROC    FAR                     ; Entry from dBASE III PLUS.
        ASSUME  CS:CODE,ES:CODE
START:  JMP     ENTRY
;===========================================================================
;
; Data area.
;
        DB      13, 'Getkey.BIN', 10, 13
        DB      'Written by J. Mike Hedblom', 10, 13
        DB      '(C) Copyright 1987, Ashton-Tate', 10, 13
        DB      26
NDXBUFF DB      SIZE DUP (0)            ; Index file buffer.
INBUFF  DB      KEYLEN DUP (0)          ; Buffer for variable data.
HANDLE  DW      0                       ; Token from DOS for input file.
;===========================================================================
;
; Get the address of the passed dBASE III PLUS variable
; and save it for later.
;
ENTRY:  PUSH    DS                      ; Save the dBASE data segment.
        PUSH    BX                      ; Save the dBASE variable address.
;===========================================================================
;
; Get the filename from variable that was passed and put it into
; a buffer.
;                       DS:BX <- 1st byte of variable
;
        PUSH    CS
        POP     ES                      ; Set up ES=CS.
        MOV     SI,BX                   ; Load variable offset.
        MOV     DI,OFFSET INBUFF        ; Load buffer offset.
```

```
        SUB     CH,CH
        MOV     CL,BYTE PTR [BX-1]      ; Move byte from variable
        REP     MOVSB                   ;    string to buffer.
        PUSH    CS                      ; Change the data segment for
        POP     DS                      ;    the rest of the program.
;=============================================================================
;
; Extract the filename only from the buffer. The filename
; is terminated with an ASCII SPACE (20h).
;
        MOV     SI,OFFSET INBUFF        ; Buffer address.
        MOV     CX,KEYLEN               ; Get KEYLEN characters.
GET_AGAIN:
        LODSB                           ; Read a character.
        CMP     AL,SPACE                ; Is it the end of name?
        JE      GET_FOUND               ;    Yes. Go on.
        LOOP    GET_AGAIN               ;    No. Get next character.
        JMP     ERROR                   ; The filename is too long.
GET_FOUND:
        AND     AL,0                    ; Terminate the filename
        MOV     [SI],AL                 ;    with an ASCII null.
;=============================================================================
;
; Open the index file with DOS function 3Dh.
;
        MOV     DX,OFFSET INBUFF        ; Load the filename address.
        AND     AL,0                    ; AL = 0 for read only.
        MOV     AH,3Dh
        INT     21h                     ; The file handle is returned in AX,
        MOV     HANDLE,AX               ;    save it for later.
                                        ; CY is set if an error occurred.
        JNC     READ_FILE               ; Opened input, go on.
                                        ; Open of file failed,
        JMP     ERROR                   ;    error and exit.
;=============================================================================
;
; Read the index file with DOS function 3Fh.
;
READ_FILE:
        MOV     BX,HANDLE               ; Load the file handle.
        MOV     CX,SIZE                 ; Read the first SIZE bytes only.
        MOV     DX,OFFSET NDXBUFF       ; Index key buffer area.
        MOV     AH,3Fh                  ; Open the file.
        INT     21h
        OR      AX,AX                   ; Was anything read in?
                                        ; CY is set if an error occurred.
        JNZ     READ_KEY                ; Yes, go on.
                                        ; Read of file failed,
        JMP     ERROR                   ;    error and exit.
;=============================================================================
;
; Read the index key from NDXBUFF back into the input buffer.
;
READ_KEY:
        MOV     SI,OFFSET NDXBUFF       ; Start of index file.
        ADD     SI,24                   ; Start of index key in the file.
        MOV     DI,OFFSET INBUFF
        MOV     CX,KEYLEN               ; Keep track of the number of
                                        ;    characters read for later.
READ_KEY_AGAIN:
```

```
        LODSB                           ; Read a character.
        CMP     AL,NULL                 ; Is it the end of the filename?
        JE      PAD_BUFFER              ; Yes, exit.
        STOSB                           ; No, swap it and get the next.
        LOOP    READ_KEY_AGAIN
;=============================================================================
;
; If an error occurs, place the error character into the return
; variable. Reset CX to KEYLEN - 1 to pad the variable with spaces.
; The error character should be greater than ASCII 128 so that it
; won't be confused with an actual index key.
;
ERROR:  MOV     AL,ERRCHAR              ; Place the error character
        MOV     INBUFF,AL               ;     into the return variable.
        MOV     CX,KEYLEN - 1
        MOV     DI,OFFSET INBUFF        ; Reset DI to pad all but the
        INC     DI                      ;    first character with space.
;=============================================================================
;
; Pad the return variable with ASCII space and write the index key
; expression back to the original dBASE III PLUS variable.
;
PAD_BUFFER:                             ; CX = the number of spaces to pad.
        MOV     AL,20h                  ; Pad rest of variable with
        REP     STOSB                   ;    character space.
        MOV     SI,OFFSET INBUFF        ; Prepare to write the key back to
        POP     DI                      ;    the dBASE variable.
        POP     ES                      ; Restore dBASE segment to ES.
        MOV     CX,KEYLEN               ; Swap the key back to
        REP     MOVSB                   ;    the variable.
        RET                             ; Return to dBASE III PLUS.
GETKEY  ENDP
;=============================================================================
CODE    ENDS
        END     START
```



# A Hypertext System for Framework II



By Roger Wegehoft

The word *hypertext* appears to have been coined by computer enthusiast **Ted Nelson**. He defines it as a form of writing that branches on request and is best presented on computer display screens. The result is a three-dimensional document, where the reader can not only proceed in a linear fashion from start to finish but also jump to related text at key points. In practice, this technique allows a person to follow various "lines of thought" that are signaled by specially marked **keywords** or phrases in a text or graphic material on a computer screen.

According to Nelson's original description of hypertext, the user would select and activate the keyword or phrase in some fashion. This would then cause a linked block of text to be displayed. This process could continue as many times as required, with keywords or phrases expanding into other relevant blocks. When users had proceeded as far as they cared to or when they ran out of places to go, they could then retrace their steps, contracting the individual blocks of text, until they returned to their starting point. There are three short hypertext entries on the first page of this article.

Because of its object-oriented structuring of data, Framework II lends itself well to this hypertext concept. This article will show you how to implement a rudimentary hypertext system using FRED code. This will allow you to **experiment** with hypertext in the realm of your own work within the Framework II environment.

## Ted Nelson

Theodor H. Nelson — B.A., philosophy, Swarthmore; graduate study U. of Chicago; M.A., sociology, Harvard. Mostly self-taught in computers. Member of editorial board, *Computer Decisions* magazine; listed in New York Times' *Who's Who in Computers*; member of Association for Computing Machinery since 1964.

## keywords

In this Framework II hypertext system, words or phrases in boldface are used to indicate the presence of a HyperFrame reference. If you **EXTEND SELECT (F6)** over the word or phrase and select Expand from the <Alt-H> macro, the material associated with it will be ZOOMed into view. Selecting the Contract option from the same macro will return you to the point of origin. The "bold" attribute is automatically given to a new HyperFrame keyword or phrase when it is created.

## experiment

If you want to learn even more about hypertext, you can consult Ted Nelson's original definition of the concept in a book titled *Computer Lib/Dream Machines*. A revised and expanded edition of this book is, coincidentally, being reissued under the Tempus imprint of Microsoft Press this month. The ISBN number is 0-914845-49-7 and the price is $18.95.

## System Overview

These are the basic system components:

- The HyperFrame macro
- Some auxiliary functions
- The Hyperstack frame structure

The HyperFrame macro is controlled by a menu of keystroke options for expanding and contracting imbedded HyperFrame references. Also included are options for creating new frame links and for toggling Framework II's **ZOOM (F9)** ON and OFF.

Expanding a HyperFrame reference consists of three steps. First, the selected text is trimmed of trailing spaces and checked for unwanted hidden characters that might cause problems in a frame reference. Next, the cleaned-up text is used as the argument of an **@setselection** command. If the text selection is a valid frame reference, the current frame (which will be the return frame) is pushed onto the Hyperstack. The system then navigates to the text selection and positions the cursor inside the frame. Finally, depending on whether **ZOOM (F9)** is ON or OFF, the frame will be displayed as either full screen or windowed.

Contracting a HyperFrame reference pops the most recent return frame reference from the Hyperstack and, depending on the value of the Zoom_it frame, displays the return frame accordingly.

Each time a new keyword or phrase is highlighted and the Create option is chosen, a new word frame is created at the same level as the frame that contains the reference. This sibling frame is then labeled with the highlighted keyword or phrase. After creation and labeling, the cursor is positioned inside the linked frame—ready to accept elaborations on the current theme or related topics. If the keyword has already been used at that level, the system issues a warning beep and positions the cursor inside the previously created frame.

What if you want different keywords to reference the same chunk of text? In that case, the system allows for an indirect frame reference to be added to a linked frame. This is done by positioning to the linked frame's label and, after pressing **Spacebar** to edit the frame's label, pressing **ZOOM (F9)**. This places you in a full-screen edit mode where you will be able to add your second keyword as an indirect frame label reference to the second line of the frame label. Afterward, any future expansion of the second keyword will take you to that linked frame .

The auxiliary functions do the following:

- Trim: trims trailing spaces
- Translate: replaces one character with another
- Hyperpop: pops a frame reference from the Hyperstack
- Hyperpush: pushes a frame reference on the Hyperstack
- Performkey: handles switching to and from ZOOM mode when necessary

Finally, it is possible and sometimes desirable to manipulate the Hyperstack directly. By manipulate, I mean to examine or delete entries on the Hyperstack. Since the Hyperstack is like a road map of our travels through the HyperFrame document, we can use it to return quickly to any given point without having to backtrack through every frame. **Scroll Lock** will toggle between a document on the desktop and the Library. For this reason, it's a good idea to place the Hyperstack at the bottom of your Library so that the stack will always be displayed.

## Installation

To install the HyperFrame macro, follow these steps:

1. Create an outline frame like the following, with **Frames:Number Labels** ON:

```
{Alt-H} ;HyperFrame system.
    1.1  Expand
    1.2  Contract
    1.3  Make
    1.3.1  Text Link
    1.4  Options
    1.4.1  Zoom
    1.5 Quit
```

2. In the formula area of {Alt-H}, place the following FRED code:

```
@menu([{Alt-H}])
```

3. In the content area of the frame labelled 1.1 Expand place the following text:

```
Expand HyperFrame (Thread Forward)
```

4. In the formula area of the 1.1 Expand frame place the FRED code for the Expand function. (**Note:** Unless otherwise specified, all the FRED code for these macros and functions is listed at the end of this article.)

5. In the content area of the frame labelled 1.2

Contract place the following text:

```
Contract HyperFrame (Thread Backward)
```

6. In the formula area of the 1.2 Contract frame place the FRED code for the Contract function.

7. In the content area of the frame labelled 1.3.1 Text Link place the following text on one line:

```
Create a HyperFrame cell from the cur-
rently highlighted selection
```

8. In the formula area of the 1.3.1 Text Link frame place the FRED code for the Text Link function.

9. In the content area of the frame labelled 1.4.1 Zoom place the following text:

```
Toggle Zoom mode ON/OFF
```

10. In the formula area of the 1.4.1 Zoom frame place the FRED code for the Zoom function.

11. In the content area of the frame labelled 1.5 Quit, place the following text:

```
Exit HyperFrame macro
```

12. In the formula area of the1.5 Quit frame place the following FRED code:

```
@quitmenu
```

13. Create six empty word frames and label them as follows:

```
- Hyperpush   ;HyperFrame stack push macro
- Hyperpop    ;HyperFrame stack pop macro
- Trim        ;Trim trailing blanks
- Translate   ;Translate characters
- Performkey  ;Switch between ZOOM modes
- Zoom_it     ;Storage for ZOOM preference
```

14. In the formula area of the Hyperpush frame, place the FRED code for the Hyperpush macro.

15. In the formula area of the Hyperpop frame, place the FRED code for the Hyperpop macro.

16. In the formula area of the Trim frame, place the FRED code for the Trim macro.

17. In the formula area of the Translate frame, place the FRED code for the Translate macro.

18. In the formula area of the Performkey frame, place the FRED code for the Performkey macro.

19. Place #TRUE in the formula area of the Zoom_it frame with #TRUE.

20. Create a word frame named Hyperstack and move all six of the frames that you've just created into it.

21. Construct an outline frame with one containing frame in the following form:

```
Hyperstack
    - _bottom-of-stack
```

22. Move the HyperFrame frame structure {Alt–H} into the Macros section of your Library.

23. Copy the Hyperstack structure to the bottom of your Library.

This completes the installation of the HyperFrame system.

## Testing

To test the HyperFrame macro, create a new word frame on the desktop and label it "HYPER". Now press **Ctrl-DownLevel** to create another frame within that one, which you can label Main. (We place the Main frame inside the HYPER frame so that subsequently created frames will all be at the same level; this is an initial navigation requirement for our macro.) Highlight the Main frame and perform a **ZOOM (F9)** on this frame. You should now be inside the content portion of the Main frame. Type the following sentence:

```
This is a test.
```

**EXTEND SELECT (F6)** the word "test" and while it is highlighted execute the {Alt-H} macro. From the HyperFrame menu, which should appear at the bottom of your screen, select Make and then Text Link. After a moment you should be in a ZOOMed empty word frame. Type the following:

```
Here we are in a frame called "test."
```

At this point, execute the {Alt-H} macro again and select Contract. You should find yourself back where you started in the Main frame. You have just created your first HyperFrame link. Move the cursor and note that the word *test* is now in boldface, indicating that it is a HyperFrame keyword.

Now try the HyperFrame macro one more time and this time select the Expand option after highlighting the word *test.* You should again find yourself in the "test" frame. Choosing Contract again should return you to Main. Note you could have continued to create new references from within already expanded frames and more from within those and so forth. The only restrictions are computer memory, disk space, and the maximum allowable unique frame name references at the same level.

For an example of an "indirect" reference, manually go to the frame labeled test, press **Spacebar**, and then **ZOOM (F9)** the label. Now enter "HYPER.test2" directly underneath the word test on a line by itself. Before exiting you should see the following:

```
test
HYPER.test2
```

Hitting **Esc** should bring you back to the frame border. Now go back into the Main frame and type another line which reads:

```
Let's go to test2.
```

Highlight "test2" and create a new Text Link. After Contracting back to Main, highlight "test" again and Expand it. You should now find that "test2" is also linked to the test frame.

## Future Directions

The implementation of hypertext presented here attempts to remain faithful to the basic seed of inspiration. While adapting the hypertext concept to the Framework II environment, I was reminded of a passage from Lewis Carroll's *Through The Looking Glass*. It goes something like "... it means exactly what I wish it to mean. Nothing more and nothing less." In other words, many features not supported in this interpretation of hypertext could be implemented easily. For example, the present system allows only one-to-one relationships between nodes. Many-to-one, one-to-many, and many-to-many relationships would be desirable, along with natural ways of selection and traversal that are transparent to the user. HyperFrame nodes could also be designed to perform auxiliary functions or programs as by-products of traversing the frames.

Applications abound for this type of information retrieval. This is new territory and has yet to be explored properly (at least in the context of Framework II) but the promise of a new level of utility for both the author and the reader is there.

---

**[{Alt-H}].Expand**

```
; Expands a HyperFrame entry.
;
@local(current, cr, ret_frame),
@echo(#OFF),
cr := @chr(13),
current := @Trim(@textselection),
@if(@len(current) > 1,
    @list(
        @if(@scan(cr, current) <> #N/A!,
            current := @Translate(cr, " ", current, 1)
        ),
        ret_frame := @setselection,
        @if(@not(@iserr(@setselection(current))),
            @list(
                @Hyperpush(ret_frame),
                @isabend(#FALSE),
                ; Notice the space before the {F9} on the following line.
                @performkeys(" {F9}{Dnarrow}"),
                @if(@isabend,
                    @performkeys("{Uplevel}"),
                    @list(
                        @performkeys("{F6}{End}"),
                        current := @Trim(@textselection),
                        @performkeys("{Uplevel}{Uplevel}"),
                        @if(@iserr(@setselection(current)),
                            @list(
                                @beep(800, 10),
                                @setselection(@Hyperpop)
```

```
                                    )
                                )
                            )
                        )
                    ),
                    @beep(800, 10)
                ),
                @Performkey
            ),
            @beep(800, 10)
        ),
        @echo(#ON),
        @quitmenu
```

---

**[{Alt-H}].Contract**

```
; Contract a HyperFrame entry
;
@local(current, ret_frame),
@echo(#OFF),
ret_frame := @Trim(@setselection),
current   := @Hyperpop,
@if(current <> #N/A!,
    @setselection(current),
    @list(
        @setselection(ret_frame),
        @beep(800,10),
        @performkeys("{Esc}{Esc}{Ctrl-Uplevel}{Return}{Scroll-Lock}")
    )
),
@Performkey,
@echo(#ON),
@quitmenu
```

---

**[{Alt-H}].Make.Text Link**

```
; Create a HyperFrame reference from the current text selection.
;
@local(cr, ret_frame, current),
@echo(#OFF),
cr := @chr(13),
;
; Test to see if the currently highlighted word is just a blank.
@if((current := @Trim(@textselection)) <> #N/A!,
    @list(
        ;
        ; If there's a CR in the selection, remove it.
        @if(@scan(cr, current) <> #N/A!,
            current := @Translate(cr, " ", current, 1)
        ),
        ;
        ; Store the return frame for the frame stack.
        ret_frame := @setselection,
        @Hyperpush(ret_frame),
        ;
        ; Check for a duplicate reference at the same level as our current
        ; frame.  If there is none, set the selection to bold, create the
        ; linked frame, otherwise beep.
```

```
        @if(@iserr(@setselection(current)),
            @performkeys("{Ctrl-W}B{Uplevel}{Ctrl-C}E" & current &
                         "{Return}{F9}{Ctrl-W}A{Uplevel}"),
            @beep(800,10)
        ),
        ;
        ; Correct for current "zoom" state.
        @Performkey,
    ),
    ;
    ; If the selection is blank, beep.
    @beep(800, 10)
),
@echo(#ON),
@quitmenu
```

---

**[{Alt-H}].Options.Zoom**

```
; Select the correct frame, and zoom it if necessary.
;
@local(current, mac, ret_frame),
@echo(#OFF),
ret_frame := @setselection,
Zoom_it   := @not(Zoom_it),
@setselection(ret_frame),
@Performkey,
@echo(#ON),
@quitmenu
```

---

**[{Alt-H}].Hyperpop**

```
@local(length, ret_frame),
@setselection("Hyperstack"),
@performkeys("{Dnlevel} {F6}{Ctrl-End}"),
ret_frame := @textselection,
length    := @len(ret_frame),
ret_frame := @mid(ret_frame, 2, length - 2),
@if(ret_frame <> "bottom-of-stack",
    @performkeys("{Esc}{Esc}{Del}"),
    ret_frame := #N/A!
),
@return(ret_frame)
```

---

**[{Alt-H}].Hyperpush**

```
@local(mac, ret_frame),
ret_frame := @item1,
@setselection("Hyperstack"),
@performkeys("{Dnlevel}{Ctrl-C}E_" & ret_frame & "{Return}{Uplevel}"),
@setselection("Library"),
@performkeys("{Scroll-lock}"),
@return(@item1)
```

**[{Alt-H}].Translate**

```
;@Translate(target, replacement, source, starting-position)
;
; Translate all occurrences of target string in source string beginning
; at starting position with replacement.
;
@local(target, replacement, source, position, loc, length),
@if(@and(@item1 <> #NULL!, @item2 <> #NULL!, @item3 <> #NULL!),
    @list(
        length      := @len(@item3),
        target      := @item1,
        replacement := @item2,
        source      := @item3,
        @if(@item4 <> #NULL!,
            position := @item4,
            position := 1
        ),
        @while((loc := @scan(target, source, position)) <> #N/A!,
            @if(loc > 1,
                source := @mid(source, 1, loc - 1) & replacement &
                          @mid(source, loc + 1, length - loc),
                source := replacement & @mid(source, 2, length - loc)
            )
        ),
        @return(source)
    ),
    @return(#N/A!)
)
```

---

**[{Alt-H}].Performkey**

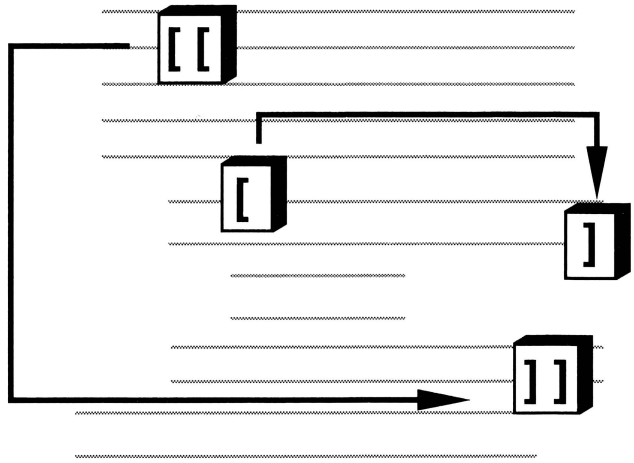
```
; Performs correct keystroke, depending on the state of the Zoom_it frame.
;
@if(Zoom_it,
    @performkeys("{F9}"),
    @performkeys("{Dnlevel}")
)
```

---

**[{Alt-H}].Trim**

```
; Trims trailing white space from character strings.
;
@local(string, length),
@if(@item1 = " ",
    @return(#N/A!),
    @list(
        string := @item1 & " ",
        length := @len(string),
        @while(@and(length > 1, @mid(string, length, 1) = " "),
            length := length - 1
        ),
        @return(@if(length <= 1,
                    #N/A!,
                    @mid(string, 1, length)
                )
        )
    )
)
```

# Finding Matching Parentheses



By Quinn Wildman and Kenneth N. Getz

FRED, like many block-structured programming languages, uses parentheses or braces as a major element. As every beginning programmer learns (usually the hard way), every opening parenthesis must be matched by exactly one closing parenthesis. Although careful indentation can help immensely in making sure that all parentheses match up, even this is often not enough when an extra parenthesis is found.

Errant parentheses will trouble you no more, however, if you use the programs presented in this article. These programs can help pinpoint the mate of a currently highlighted parenthesis, brace, or bracket. We've presented two versions of {Alt-P}—one in FRED only and another in C for owners of the *Framework II Developer's Toolkit.* The FRED version makes judicious use of **Ctrl-Rightarrow** to work through text, but since it must search through every word or character in an interpreted loop, it tends to be a bit slow. The C version is much faster and contains some other benefits as well, but it's more complicated to set up. Owners of the *Framework II Developer's Toolkit* will no doubt opt for this faster version, even though it requires more work. (The C version of the macro was written and tested with Microsoft MASM version 4.0 along with both Microsoft C version 4.0 and Turbo C version 1.0. Users of other C compilers may need to make changes in the code and compilation instructions to make it work with their compiler.)

## How {Alt-P} Works

Both versions of {Alt-P} follow the same basic algorithm in searching for matches. Both are written to be used as macros (stored in the Library Cabinet) and are called into action by pressing **Alt-P** while the cursor is highlighting a parenthesis "(" or ")". The C version also checks for braces "{" or "}" and brackets "[" or "]".

In both versions, if the macro is invoked when the cursor is not on a legitimate character, you hear a beep and get an error message. If the cursor is on a correct character, the macro "walks" through the frame, looking for the character's mate. If the cursor is on an opening character, the macro walks forward in the frame, and if the cursor is on a closing character, the macro walks backward looking for a match.

During the search, the macro keeps a counter of how many characters it finds matching the original character. Every time it finds the same character, it adds one to the counter, and every time it finds the correct target character, it subtracts one from the counter until it finds a target character and the counter is 0. If it reaches the end of the frame before this happens, an error has occurred, and the macro will indicate this. If a match is found, the cursor is left on the matching character. In both programs, if a quote (") is encountered, all calculations are suspended until the macro finds a match-

ing quote. This allows you to embed parentheses inside quoted strings that won't affect the search.

Neither macro will make a special case of comments, however. Checking to find out whether or not the current position was in a comment would significantly slow the operation of the macro, so this ability was left out. This won't cause a problem unless you use unquoted, unmatched parentheses within your comments. That would adversely affect the matching process.

## Setting Up

No matter which version of this macro you use, you'll need to create a new word frame in the Macro section of your Library Cabinet. We've chosen to name the macro {Alt-P}, though you could use any **Alt**-key combination you wish, as long as you change all references in the program code accordingly.

## The FRED Version

To set up the FRED-only version of this macro, type the program code in the formula area of the word frame you created in your Library. When finished, be sure to save the Library Cabinet.

## The C Version

### Entering the Program Code

To create the version of the macro using the *Framework II Developer's Toolkit*, you will need two external files. You should first make a copy of Mymain.ASM from the \LLI\RDTXT directory of your Framework II Developer's Toolkit Disk #2, name it Pmain.ASM and make the changes outlined in the article "Using the *Framework II Developer's Toolkit* with Microsoft C" in the June 1987 issue of ***TechNotes***. (If you don't have the June 1987 issue, call 800-331-4164 in California or 800-828-2514 in other states, for information on ordering back issues). If you have a previously altered version of this file, just make a new copy and start from there.

Once the alterations have been made to Pmain.ASM, you should modify the first portion to match the listing for Pmain.ASM given in this article. You should then type in the code for Parens.C as listed here. To complete the setup, you should create two empty word frames inside the {Alt-P} frame you created in your Library Cabinet so the structure looks like this:

```
- {Alt-P}
     - Main
     - Value
```

Enter the appropriate program code into the formula areas of {Alt-P} and [{Alt-P}].Main and leave [{Alt-P}].Value empty.

### Compiling

Before compiling or assembling these files, you must have installed all assemblers and compilers according to their instructions. Make sure at this point that all programs have been installed correctly and are either in the current directory or in the DOS PATH.

1. To assemble Pmain.ASM, type:

   ```
   MASM Pmain;
   ```

2. To compile Parens.C, type:

   **Microsoft C:**

   MSC Parens /Gs;

   (The /Gs disables stack checking.)

   **TurboC:**

   TCC -c Parens

   (Turbo C uses no stack checking by default.)

3. To link all the modules, type:

   **Microsoft C:**

   LINK Pmain + Parens, Parens /NOD;

   (The /NOD leaves out the default libraries.)

   **Turbo C:**

   TLINK Pmain Parens, Parens

   (Turbo C leaves out default libraries unless told otherwise.)

   In either case, this creates a file called Parens.EXE

### Putting It Together

Once Parens.EXE has been created, go back to {Alt-P} in your Library Cabinet and use the **Disk:Get File by Name** menu option to bring in Parens.EXE. This completes the building of your macro.

**{[Alt-P]}**

```
; Parentheses checking macro.
;   by Quinn Wildman
;   December 1, 1987
;
@if(@and(@textselection <> "(", @textselection <> ")"),
    @list(
        @eraseprompt,
        @beep,
        @prompt("Highlight a ) or ( before pressing Alt-P"),
        @return(1)
    )
),
@local(cnt, inquote),
cnt := 0,
@isabend(#FALSE),
inquote := #FALSE,
@if(@textselection = "(",
    @list(
        @performkeys("{Rightarrow}"),
        @while(@not(@isabend),
            @if(@textselection = ")",
                @list(
                    @if(@not(inquote),
                        @if(cnt = 0,
                            @return(0),
                            cnt := cnt - 1
                        )
                    ),
                    @performkeys("{Rightarrow}")
                ),
                @if(@textselection = "(",
                    @list(
                        @if(@not(inquote),
                            cnt := cnt + 1
                        ),
                        @performkeys("{Rightarrow}")
                    ),
                    @if(@and(@or(@and(@textselection >= "0", @textselection <= "9"),
                                 @and(@textselection >= "A", @textselection <= "Z"),
                                 @and(@textselection >= "a", @textselection <= "z")
                             ),
                             @not(@isbop(@textselection))
                        ),
                        ;Consider possibility of single character before " or )".
                        @list(
                            @performkeys("{Rightarrow}"),
                            @if(@and(@textselection <> """",
                                     @textselection <> ")",
                                     @not(@isbop(@textselection))),
                                @performkeys("{Leftarrow}{Ctrl-Rightarrow}" &
                                             "{Rightarrow}")
                            )
                        ),
                        @list(
                            @if(@textselection = """",
                                inquote := @not(inquote)
                            ),
                            @performkeys("{Rightarrow}")
```

```
                    )
                )
            )
        )
    ),
    ;Must be a ")"
    @list(
        @performkeys("{Leftarrow}"),
        @while(@not(@isabend),
            @if(@textselection = "(",
                @list(
                    @if(@not(inquote),
                        @if(cnt = 0,
                            @return(0),
                            cnt := cnt - 1
                        )
                    ),
                    @performkeys("{Leftarrow}")
                ),
                @if(@textselection = ")",
                    @list(
                        @if(@not(inquote),
                            cnt := cnt + 1
                        ),
                        @performkeys("{Leftarrow}")
                    ),
                    @if(@and(@textselection >= "-",
                             @textselection <= "z",
                             @textselection <> "<",
                             @textselection <> ">"),
                        ;Consider possibility of single character before " or ("
                        @list(
                            @performkeys("{Leftarrow}"),
                            @if(@and(@textselection >= "-",
                                     @textselection <= "z",
                                     @textselection <> "<",
                                     @textselection <> ">"),
                                @performkeys("{Rightarrow}{Ctrl-Leftarrow}" &
                                             "{Leftarrow}")
                            )
                        ),
                        @list(
                            @if(@textselection="""",
                                inquote := @not(inquote)
                            ),
                            @performkeys("{Leftarrow}")
                        )
                    )
                )
            )
        )
    )
)
```

**Pmain.ASM**

```
            EXTRN  _parens:NEAR
_DATA       SEGMENT        WORD  PUBLIC 'DATA'
_DATA       ENDS
CONST       SEGMENT        WORD  PUBLIC 'CONST'
CONST       ENDS
_BSS        SEGMENT        WORD  PUBLIC 'BSS'
_BSS        ENDS
DGROUP      GROUP  CONST, _BSS,  _DATA, _STACK
_DATA       SEGMENT        WORD  PUBLIC 'DATA'
PUBLIC      FW_EXEC
fw_exec           DD     0                  ; Will be plugged by Framework II.
proc1       DW     OFFSET parensf            ; Offset address of procedure # 1.
dummy_buffer DW    0                   ; Used by num_byte.
PUBLIC __acrtused                      ; Trick to force in startup code.
__acrtused = 9876h                     ; Funny value not easily matched in SYMDEB.
_DATA       ENDS

_STACK SEGMENT       WORD  STACK  'DATA'
            DW     1024 dup (?)        ; Declare stack for local usage.
_STACK      ends
PGROUP GROUP  _TEXT
_TEXT  SEGMENT             BYTE   PUBLIC 'CODE'
ASSUME CS:_TEXT,DS:DGROUP,ES:DGROUP,SS:DGROUP

parensf proc  far
        call  _parens
        ret
parensf           endp

; The rest of the program is to be left unchanged.
```

---

**Parens.C**

```
/*************************************************************************/
/*  File.....: Parens.C                                                  */
/*  Author...: Kenneth N. Getz                                           */
/*  Date.....: December 1, 1987                                          */
/*  Notice...: 1987, All rights reserved.                                */
/*  Notes....: Parens.C finds a mate for a given brace, bracket, or      */
/*             parenthesis.                                              */
/*                                                                       */
/*             Compiled and tested with Microsoft C v4.0 and Turbo C v1.0 */
/*             It should work fine with any other C compiler compatible  */
/*             with the Framework II Developer's Toolkit.                */
/*                                                                       */
/*             Returns:                                                  */
/*                0 : No error                                           */
/*               -1 : Too few characters in the frame.                   */
/*               -2 : Current character not in "(){}[]"                  */
/*               Any positive number: Count of mismatched characters.    */ /
*************************************************************************/
#define FALSE 0
#define TRUE !FALSE
#define FORWARD 1
#define REVERSE -1
parens()
{
```

```
int counter;                    /* Counter for matching parentheses, etc. */
int direction;               /* Indicates direction -- FORWARD or REVERSE. */
int end;                    /* Ending position of marked range.  Not used. */
int inquote;                        /* Test -- currently in quoted string? */
int pos;                                        /* Index into the frame. */
int start;                /* Starting position of marked region in frame. */
char ch;                    /* All characters besides first read into ch. */
char first;           /* The first character read.  Should be in "(){}[]" */
char target;              /* Calculated target character, based on first. */

if (NUM_BYTE(0,0) < 2)        /* If there are less than two characters in */
   return(-1);                                    /* the frame, quit now. */
inquote = FALSE;                                  /* Initialize variables. */
counter = 1;
RTEXTSEL(0, &start, &end);             /* Find out where we currently are. */
pos = start;                     /* Store starting position for later use. */
FW_READ(0, 0, 1, pos, &first);               /* Get the current character. */
switch (first)            /* Set up for searches based on character read. */
   {
   case '(':
      direction = FORWARD;
      target = ')';
      break;
   case ')':
      direction = REVERSE;
      target = '(';
      break;
   case '{':
   case '[':
      direction = FORWARD;
      target = first + 2;                         /* Target is '}' or ']'. */
      break;
   case '}':
   case ']':
      direction = REVERSE;
      target = first - 2;                         /* Target is '{' or '['. */
      break;
   default :                /* If character not legal, return error code. */
      return(-2);
   }
pos += direction;                               /* Go on to next character. */
while (FW_READ(0,0,1,pos, &ch))          /* As long as we get a character, */
   {
   if (ch == '\"')                                    /* if we got a quote, */
      inquote = !inquote;                     /* reset the quote indicator. */
   if (!inquote)                /* If we're not currently inside a quote, */
      if (ch == first)         /* if we found another beginning character, */
         counter++;                               /* increment the counter. */
      else if (ch == target)           /* If we found an ending character, */
         {
         if (--counter == 0)                  /* decrement and check for 0. */
            {
            STEXTSEL(0, pos, pos + 1);        /* If we're done, set cursor */
            break;                                    /* and quit looking. */
            }
         }
   pos += direction;                       /* Go on to the next character. */
   }
return(counter);               /* This will return 0 if match found, or a  */
}           /* positive number indicating the number of mismatches if not. */
```

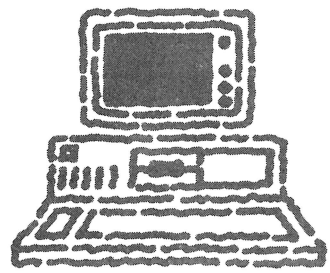
**[{Alt-P}]**

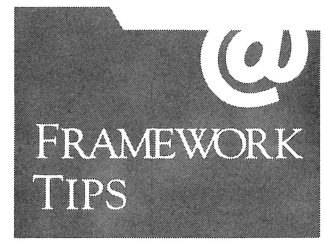
```
;
;    by Kenneth N. Getz ;
;    December 1, 1987 ;
@[{Alt-P}].Main
```

---

**[{Alt-P}].Main**

```
Value := @exec(Parens, 1),
; Repaint the screen.
@echo(#ON),
@if(Value = 0,
    @return(0),
    @list(
        @beep,
        @eraseprompt,
        @if(Value = -1,
            @prompt("The frame needs to be larger."),
            @if(Value = -2,
                @prompt("Highlight a ) or ( before pressing {Alt-P}."),
                @if(Value = 1,
                    @prompt("There was " & @integer(Value) & " mismatched pair."),
                    @prompt("There were " & @integer(Value) & " mismatched pairs.")
                )
            )
        )
    )
)
```






## Extended Memory

By Chris LoVasco

In Framework II, the same file can use different amounts of memory depending on whether extended memory is used and how much extended memory is defined for use in the Setup program.

The basic unit of data in Framework II is a frame. Desktop frames contain multiple internal frame structures. These internal frame structures vary in size depending on the amount of information they contain. When a single desktop frame is loaded into real memory (640K), these internal frame structures are stored contiguously so there is no wasted memory. However, because extended memory is not directly addressable, frame structures are stored in a different manner when it is used.

Extended memory is divided up into data blocks. Each data block can contain only a single frame structure, but a single frame structure can occupy more than one data block. If a frame structure is smaller than the data block that contains it, the difference is wasted memory. This wasted memory causes the same file to use different amounts of memory on the desktop depending on the amount of extended memory used.

The size of the data block is calculated and set when Framework II loads and initializes extended memory. The larger the total extended memory, the larger the data block size, and the more memory is wasted for small frame structures. The number of data blocks contained in extended memory is less than 32,000 if the total extended memory is less than 512K (see formula). Framework II uses one of the following formulas to determine data block size, depending on the amount of extended memory defined.

**Maximum extended memory size < 512K:**

- Block Size = 16 bytes
- Number of Data blocks = integer(Maximum Extended Memory Size / Block Size)

**Note:** The smallest block size allowed by Framework II is 16 bytes.

**Maximum extended memory size >= 512K:**

- Data Block size = Maximum Extended Memory Size / 32,000

- Number of Data blocks = integer(Maximum Extended Memory Size / Block Size)

**Note:** Block Size is rounded up to the nearest multiple of 16 bytes. For example,

1537K / 32000 = 48.03 bytes

would be rounded up to a block size of 64 bytes. The total extended memory is then divided by the rounded block size to calculate the total number of data blocks.

## Forcing Frames Open or Closed

FRED programs sometimes need to force frames on the desktop to be in the proper state before they can proceed. For example, a program might be written that depends on a database frame in forms view. The following code accomplishes this:

```
@while(@frameview<>2,
          @performkeys("{F10}"))
```

FRED continually presses **VIEW (F10)** until **@frameview** returns 2, which means the database is in forms view.

There is no FRED function that tests a frame on the desktop to see whether it is open or closed. It is, however, quite simple to force a frame to be open or closed. @Openframe is a user-defined function to perform this task. The first parameter is the name of the frame, given as a string. The second parameter is #TRUE to open the frame, or #FALSE to close it. For example,

```
@Openframe("testframe", #FALSE)
```

will force "testframe" to be closed.

## Clearing Results of Calculations

There are times when a Framework II user would like to hide the results of a calculation without deleting the formula itself. For example, you can set up a spreadsheet template that contains formulas but no actual data. Cells that contain formulas referencing empty cells display #N/A! until they are recalculated with valid data.

A cell always displays the result of the last evaluation in its formula. {Alt-H} is a macro that uses this fact, and properties of Framework II's formula editor, to blank the display of a cell. It modifies and recalculates the cell's current formula and then deletes the modification without recalculating, so the cell's true formula is not evaluated.

```
; @Openframe.FW2
; by Kenneth N. Getz
; December 1, 1987
;
; Usage: @Openframe(framename, open)
;
;        framename  ::=  character string
;        open       ::=  logical constant
;                        #TRUE  -- open frame
;                        #FALSE -- close frame
;
;        Returns 0 if the frame is available, -1 if it is not.
;
@echo(#OFF),
@if(@iserr(@setselection(@item1)),
    @list(
        @echo(#ON),
        @return(-1)
    )
),
; Open the frame.
@performkeys("{F4}{Esc}");
; If second parameter is #FALSE, close the frame.
@if(@not(@item2),
   @performkeys("{Return}")
),
@echo(#ON),

@return(0)
```

Enter the code into the formula side of a frame labeled {Alt-H} and move the frame into your Library. To use the macro, navigate to the cell you wish to blank and press **Alt-H**.

**[{Alt-H}].FW2**

```
; by Scott Brown
; December 1, 1987
;
; Macro to clear a cell's contents without
; deleting the formula.
;
@echo(#OFF),
@performkeys("{F2}@return("" ""){Return}"&
             "{F2}{Ctrl-1}{Ctrl-2}{Del}"&
             "{Esc}{Return}"),

@echo(#ON)
```

## Calculating Compound Interest

Framework II does not have a built-in function to calculate compound interest. Compound interest is the interest earned on an investment at a given interest rate and number of periods. For example, if you place $1,000 in an account at 8% annual interest, compounded monthly, the balance of the account in 3 years will be $1,270.24.

The formula for this calculation is

$$\text{principle} * (1 + \text{rate})^{\text{periods}}$$

@Compound is a user-defined function that performs the compound interest calculation. As in the formula above, @Compound calculates the value of the investment, as opposed to the interest earned.

To use @Compound, it must be on the desktop or in your Library. The interest calculation above would be entered as

```
@Compound(1000, 8% / 12, 36)
```

Note that the annual interest rate is divided by the number of periods in a year. If you forget to do this the result will reflect interest compounded once annually for "n" years.

**;@Compound.FW2**

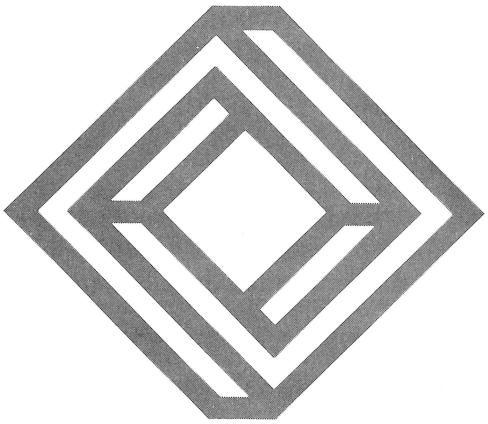
```
; by Chuck Litzell
; December 1, 1987
;
; Returns value of an investment compounded
; at an interest rate
; for a given number of periods.
;
; @compound(principle, rate, periods)
;
@return(@item1 *  (1 + @item2) ^ @item3)
```

## Comment/Uncomment Blocks

The macro printed here makes it easy to comment or uncomment blocks of FRED code. Normally, you would have to move manually to the beginning of each line you wished to comment, press ";", and move on to the next line. This macro allows you to **EXTEND SELECT (F6)** a region, press the **Alt**-key assigned to this macro, and let the macro "do the walking." The macro will also uncomment blocks with similar ease.

To perform this task, the macro first finds the beginning and ending lines of the marked region by performing {F6} twice, which returns the beginning and ending lines of the region in **@panel2**. Once the beginning and ending lines are known, it's easy to place or remove the semicolons.

```
; The macro is a toggle.  It checks the
; first character on a line.  If it is a
; semicolon, it will be deleted. If the
; first character is not a semicolon,one
; will be inserted.
;
;   by Quinn Wildman
;   December 1, 1987
;
@local(begin, end, lines),
begin := @panel2,
@isabend(#FALSE),
@performkeys("{F6}"),
@if(@isabend,
    @list(
        end := @panel2,
        @performkeys("{Return}"),
        begin := @panel2
    ),
    @list(
        end := @panel2,
        @performkeys("{Esc}")
    )
),
lines := end - begin + 1,
@while(lines > 0,
    @performkeys("{Home}"),
    @if(@textselection = ";",
        @performkeys("{Del}"),
        @performkeys(";")
    ),
    @performkeys("{Dnarrow}"),
    lines := lines - 1
)
```



# ANOMALIES

## DEFINITION OF ANOMALY

We use the word "anomaly" to mean any one of the following:

- INCORRECT COMMAND (or function) IMPLEMENTATION. A command or function does not execute as specified.
- INCOMPLETE COMMAND (or function) IMPLEMENTATION. A command or function does not execute as specified in every use of the command or function.
- DATA OR LOGICAL BOUNDARY LIMITATION. A command or function cannot perform an operation at a specific data or logical boundary.

**NOTE:** Ashton-Tate reserves the right to withold any further information that may reveal proprietary or trade secret information.

dBASE III PLUS

## CREATE SCREEN

CREATE SCREEN creates a format file only when a database file is open while the screen file is saved. Without a database file open, dBASE III PLUS appears to save the screen format; however, neither an .SCR or .FMT file is generated. Selecting the CREATE SCREEN **Options:Generate text file image** creates an empty .TXT file when the screen format is saved.

Having a database file open in another workarea and not in the current workarea leaves the format file open when CREATE SCREEN is exited. It is necessary to QUIT dBASE III PLUS to close the format file.

To create a format screen, USE a database file in the current workarea before entering CREATE SCREEN or select a database file from the CREATE SCREEN **Set Up:Select a database file.**

## CREATE VIEW FROM ENVIRONMENT

With SET CATALOG ON, CREATE VIEW FROM ENVIRONMENT turns the status bar on. After entering the title for the view at the catalog prompt, the status bar remains and scrolls up the screen as new commands are entered. SET STATUS OFF has no effect on the display.

### Date Fields

A slash (/) entered in the month or day portion of a dBASE III PLUS date field or memory variable changes the value of that portion to 00. This produces the error message "Invalid date (press SPACE)" when exiting the field or variable. With SET CENTURY OFF, entering a slash in the year portion of the date produces no error and the year becomes 1900. With SET CENTURY ON, entering a slash in the year portion of a date changes the year to 0000 and produces the error message. dBASE III does not accept entry of any characters except numerals in date-type variables or fields.

## dBCODE

Commands that include an asterisk immediately followed by a function and not separated by a space produce the error message "Data type mismatch." [9] when dBCODEd. An example is

```
mem2 = mem2 *ASC(SUBSTR(mem1,1,1))
            ↑
            No space.
```

Inserting a space after the asterisk allows the command line to be dBCODEd correctly.

## dBCODE

When a negative value or no value is specified as the lower value in a RANGE statement, the error message "Unrecognized phrase/keyword in command." [36] occurs when the command is dBCODEd and executed. To work around this problem, include a value for the lower range. If the lower value is negative, enclose it in parentheses. For example,

```
@ 10,20 GET mem RANGE (-9),-1
```

## dBCODE

In dBASE III PLUS, a LOOP outside of a DO WHILE...ENDDO structure produces no errors when interpreted, however it does terminate routine execution like RETURN. When dBCODEd, LOOP is ignored when encountered outside of a DO WHILE...ENDDO structure, unless it is inside of an IF...ENDIF structure. However, a LOOP inside of an IF...ENDIF and not contained in a DO WHILE...ENDDO structure displays the error message "*** ENDIF unexpected in file <Filename> on line <Line #>." when dBCODEd.

## ERASE

If a PATH is followed to USE a database file, the file can be ERASEd by specifying the explicit path and filename, even though the file is open. Continuing to use the file once it has been deleted can cause lost clusters on the disk. For example,

```
SET PATH TO B:
USE Test          && File is on drive B:.
ERASE B:Test.DBF && File is ERASEd
                  && from disk.
APPEND            && Adding enough records
                  && produces lost clusters
                  && on disk.
```

## LABEL FORM TO FILE

Creating a text file with the command

```
LABEL FORM <filename> TO FILE <filename>
```

may create a text file with carriage returns that are not matched to linefeeds at the end of each record. APPENDing such a file as an SDF file inserts a blank record between each record for each unmatched carriage return.

These carriage returns correspond to the **Options:Lines between labels** choice in CREATE/MODIFY LABEL.

## ON ERROR/ESCAPE/KEY

Attempting to dBCODE the line

```
ON ESCAPE
         ↑
         There is a blank space here.
```

produces the error message "*** Illegal condition in ON statement in file <Infile> input line #." This is true of all three ON commands. Removing the extra space allows the file to dBCODE properly.

## SET DELIMITERS ON

With SET DELIMITERS ON, answering "Yes" to the "Input data now?" prompt in CREATE causes the fields to display without delimiters. However, once the file has been created, the delimiters are correctly displayed on all the full-screen commands that honor SET DELIMITERS.

FRAMEWORK II

## Framework II Advanced Topics

Page 2-34 in *Advanced Topics Framework II*, refers to using the Hayes modem ATO command to set the modem in the originate mode when changing from a voice connection to a data connection. This should refer to the ATD command, not ATO.

## Using Framework II

On page 10-17 of *Using Framework II*, the documentation states that Framework II prompts the user for the name of the resulting file when performing an export. Framework II does not prompt the user for the filename, instead, it uses the frame name and assigns the appropriate extension according to the chart on page 10-17.

## Quick Reference Guide

The description of **Shift-Tab** on page QR-9 of the *Quick Reference Guide Framework II* booklet is incorrect. The booklet states that a **Shift-Tab** moves the cursor to the next selection. **Shift-Tab** moves to the previous selection.

## Moving Active Telecomm Frame

In Framework II, using **MOVE (F7)** to move an active telecommunications frame into a file Cabinet, locks the system when the **Esc** key is pressed after displaying the message "Active Telecomm frame has been deleted and session ended (Press **Esc** to continue)." This problem does not occur if the active telecommunications frame is copied with **COPY (F8)**.

To work around this problem, **End Session** before attempting to move the frame.

RAPIDFILE

## Highlighted Records

Choosing the **Layout:Design layout** option when the cursor is positioned on a highlighted record causes display problems. When the cursor is moved from field to field in this mode, multiple fields may become highlighted.

## Select Matching Records

RapidFile may return inconsistent results if a character string is used for the match criteria when selecting or locating records. This occurs when the field being matched has been added to the file structure after the initial file creation process. When this occurs, records that do not match the criteria may appear in the working set in addition to those records that do match the criteria.

To work around this problem, enclose the character string in quotation marks to force a literal search. This method properly filters the database.

## Read.ME File

The Read.ME file that is included with the 5.25-inch version of RapidFile is not included with the 3.5-inch version.

## Personalizing Memos

Under Personalizing Inserted Data in a Form Letter, on page 8-7 of *Learning RapidFile*, the documentation outlines the steps needed to personalize a name in the greeting field. Although these steps are correct, they fail to mention that once you advance to the next record with **Ctrl->**, all customization to the current memo is lost and not saved to disk. To print the customized memo, you must select the **Print:Begin printing** or **Merge:Print form letters:Current** option while the customized memo is displayed on the screen.

## BIOS Problems

The Leading Edge Model D PLUS and the NEC MultiSpeed laptop computers have a BIOS compatibility problem that appears when running RapidFile. One of the BIOS calls that RapidFile makes is not handled correctly by the BIOS used in these machines. RapidFile can be loaded, but the system locks up during the first operation performed.

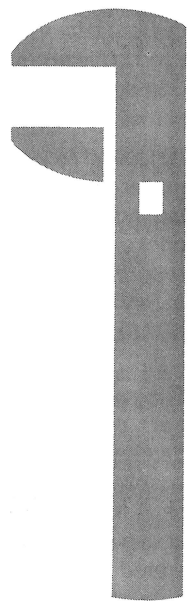
The following procedure can be used to patch the RapidFile system disk to work around the BIOS incompatibility. It is assumed that you have IDed the system disk and it contains a file called RF.COM.

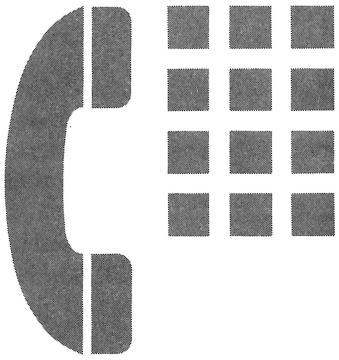
1. Put the RapidFile System disk in drive A and the DOS disk containing DEBUG.COM in drive B.
2. Using the following screen dump as a guide, proceed with the DEBUG session.

```
A>B:DEBUG RF.COM
-a 4e0c
xxxx:4E0C jmp 3ef7
xxxx:4E0F nop
xxxx:4E10 <Return>
-w
Writing 70BF bytes
-q
```

**Note:** User input is in boldface type.

After the System disk is patched, RapidFile can be run as usual from the floppy disk or installed to a hard disk using the normal installation procedures.





# ON-LINE SERVICES

Do you have a modem? Do you subscribe to CompuServe or The Source? If so, Ashton-Tate offers technical support on all of these electronic information services. You can quickly access the kind of quality technical information that you have come to expect from ***TechNotes***, and you can do it through your computer system. To take advantage of this progressive support method, you must subscribe to one of the services listed below. There are no additional charges for using our on-line support services, apart from the normal connect-time fees for each service. You can get specific information by calling one of the following numbers:

| Subscriber Service Information | | Command to find Ashton-Tate |
|---|---|---|
| CompuServe | (800) 848-8199 | GO ASHTON |
| The Source | (800) 336-3366 | ATSIG |

To give you an idea of the support that is happening on-line, the following are regular excerpts from the Ashton-Tate Forum on CompuServe. This monthly material comes from messages our Forum users have posted on the bulletin board. All message headers have been removed to protect the identities of our users. Every other message within a single conversation topic has been italicized to enhance readability.

dBASE III PLUS

## Mouse Driver Program

During a call to Ashton-Tate, one of our employees was told that someone had recently uploaded a program in this forum for a mouse driver. Apparently the techniques used in the program will be helpful in other ways as well. Can someone tell me what DL it is in and under what name?

*It's called Mouse2.ARC, and it's in DL2. I tried it and it works great, but it is not a program that you can just run and find that it works with your software. It provides the hooks for you to access the mouse from within your own program. Its main purpose is to demonstrate the use of a .BIN file to access DOS interrupts directly from within dBASE III PLUS. Give it a try, it might help you out.*

## Screen Generation

I am attempting to use dBASE III PLUS to create a database for patient information, including name of hospital, admitting diagnosis, date of admission, and so on. I have created the following field names for the above descriptions: hospital, dx, ad_date. I have generated a format file to more easily display the inputted data using @SAY and @GET statements. How can I best insert into the format file a specific hospital name as the default input for the field HOSPITAL? The present line in the .FMT file reads:

```
@ 1,10 SAY "Name of hospital"
@ 1,25 GET hospital
```

However, instead of having a delimited area of 15 reverse-video spaces for input of the name of the hospital for a given patient, I wish to have a specific hospital name preinserted into the reverse-video area of input (starting at 1,25). This name can then be changed by the user if the hospital is different from the default hospital name.

*If you're using SET FORMAT TO .../APPEND, you really can't do that. You'll need to APPEND BLANK/REPLACE fldname WITH "default hospital name," then SET FORMAT TO.../EDIT NEXT 1.*

## Append & Select

How do I APPEND from a SELECTed database to another SELECTed database?

*The usual method is to:*

```
SELECT <firstfile>
COPY TO Temp <scope etc.>
SELECT <secondfile>
APPEND FROM Temp
```

## Basic Printing

I am listing databases in a standard report format. Each report is of variable length. How do I get dBASE III PLUS to reliably go from the end of the variable report to the next perforation and to reset itself for the beginning of the next report? I'm generating 11 variable-length reports with a DO...WHILE loop, but the printer's giving me fits. I've done a good bit of homework, but the technique's eluding me.

*If you are using the same length of paper for all the reports, then once you've told the printer the page length (default is 11-inches long for most printers) just use the EJECT command to move to the top of the next form. You should also make sure that the printer is initialized correctly (the paper's loaded to the top of a form) when you turn it on.*

## Burying DOS Messages

We have a billing program for a client, one segment of which does part of the backup process at the end of each month by copying activity files to a new subdirectory and then zapping the original activity files to prepare them for use the next month. To make the new subdirectory, the program uses the command "RUN MD &subdir," and then the next lines copy 10-12 files to the new subdirectory.

The problem is this. Sometimes in the middle of the month we like to look at or print out the nicely formatted list of transactions this program produces from the files once they are in the new subdirectory. If we do run the program in the middle of the month and then again at the end, it works okay—the newly copied files just overwrite the ones we'd copied earlier, and the program ignores the command to make a new subdirectory, since the new subdirectory has already been made. Ignores, that is, except for putting the DOS message, "Unable to create new subdirectory" right on the screen to tell you something is wrong. In practice, this DOS message does not hang up the program at all. But it makes people think that the program is hung up, and so I'd like to get rid of it. I've tried SETting CONSOLE OFF just before running the MD command, and I've tried using "RUN MD &subdir > NUL," but neither approach works.

*Try this:*

```
SET CONSOLE OFF
RUN IF NOT EXIST &subdir.\*.* MD &subdir.
SET CONSOLE ON
```

*(Don't leave out the period after the first macro.) Hope this helps.*

## COUNT Function

I am trying to set up a database for complaints. Each record has the normal name and address fields and a three-character complaint code.

I want to make a summary report of the number of each type of complaint.

The COUNT command seems so slow. Is there a better way? I am currently using something like

```
Count for COMPL = 'A11' to m1
Count for COMPL = 'A12' to m2
```

then printing m1, m2, and so forth.
What would be the effect of INDEXing or SORTing on the complaint field? Does COUNT work any faster if items are grouped together? Can I count all the items in a single pass through the file?

I had also thought of indexing, then

```
DO WHILE Compl = 'A11'
       m1=m1+1
ENDDO
```

for each Compl.

*Add a dummy field to your database for totals. It needs to be numeric, and the width needs to be as wide as the largest of your totals. Then issuing the following commands will give you a database containing your totals:*

```
REPLACE ALL dummy WITH 1
INDEX ON compl TO Temp
TOTAL ON compl TO Totals FIELDS dummy
```

*Using the built-in TOTAL ON command is the easiest way to get the totals you need.*

## DBF File Format

I need the file format for .DBF files under dBASE III PLUS. I remember seeing the format somewhere, but now that I need it I can't find it. I need to convert a raw data file to a .DBF file that dBASE III PLUS can read, and I would prefer the client not have to use the IMPORT command, so I would like to build the .DBF file.

*The file format/structure is in Appendix C of* Using dBASE III PLUS, *which is volume one of the documentation for the system. The IMPORT command specifically imports PFS files. Check the documentation for APPEND FROM...SDF to see if it meets your needs.*

## Copying Records

I have a large database and am attempting to split it by using an archive database equal to the original. I want to mark for archiving and then have the record transferred to the archive file. So far the only way I can see to transfer a complete record is to read it into memory variables, mark the record for deletion, change databases, and then replace all the fields with the memory

variables. There are 109 fields in each record. This routine takes 22 seconds. Is there a better and faster method?

*If there's a field or expression that can be used to identify all the records that are to be archived (such as a date field being older than a given date), then you can use a command like*

```
COPY TO Archive FOR;
   Date_field < CTOD("08/01/87")
```

*Otherwise you'll need to create a logical field with MODIFY STRUCTURE, then change that field to .T. (true) for each record that is being archived and*

```
COPY TO Archive FOR Flag
```

*where Flag is the logical field used to identify the record being archived.*

FRAMEWORK II

## Exporting Dates to dBASE

I have a Framework II database that I am trying to transfer to a dBASE III PLUS database file. However, I am having problems getting dBASE to read my date field properly. For example, If I have the date listed as **@date**(1987,8,27), when I APPEND DELIMITED to a dBASE file it comes out garbled.

*Before you export to a dBASE delimited file from Framework II, **EXTEND SELECT (F6)** the whole date field column, and set **Integer** format in the **Numbers** pull-down menu. This produces a date format that dBASE can properly import in a delimited file. Now proceed with the export.*

## dBASE Import

I tried to import a dBASE III PLUS version of a file into Framework II and it did not work properly. All I got was the field headers but no data.

*dBASE data files import into Framework II in a two-step process, mainly because dBASE files often grow fairly large and not all the records may need to be brought into Framework II.*

*If you look at the border of the new database frame using **F2 (EDIT FORMULA)**, you'll find an **@dbasefilter()** function. Recalculating the frame without changing this formula will bring all the records that will fit in memory into the frame. You can modify the **@dbasefilter()** formula to select only the desired records by replacing the #TRUE constant with a logical equation, such as Salary > 20000. Also, you can import just a segment of the datbase if you add a third and fourth parameter which specify the first and last record numbers to import.*

*More information on this is in the FRED section of the* Framework II: Advanced Topics *book under the **@dbasefilter** function description.*

## Toshiba Keyboards

I just installed Framework II onto my new Toshiba 3100. Everything is OK, but I miss the gray + and gray – keys. I found a keyboard driver named T1200.KB on the SETUP disk and installed it, but I don't know what keys I have to press for Uplevel and Dnlevel.

*The T1200.KB file is for the Tandy 1200, not the Toshiba. The 3–1/2" format of Framework II, version 1.1, includes keyboard drivers for several laptop computers, including the Toshiba 3100. Check with Ashton–Tate Customer Service about upgrading your copy.*

## Zenith Keyboards

I am trying to run Framework II (version 1.1) on a Zenith 181. There is no keyboard driver for that particular computer in the FW package. The main problem is that the Uplevel and Dnlevel keys are extremely hard to create on the 181. Does anyone have a driver for the 181 keyboard or have any experience using a driver other than the standard IBM keyboard driver?

*This is a problem on lots of laptops. Perhaps you could just assign key macros to perform those keystrokes. For example, create a word frame in the Library Cabinet named {Alt-U}, and press **F2** to enter the formula*

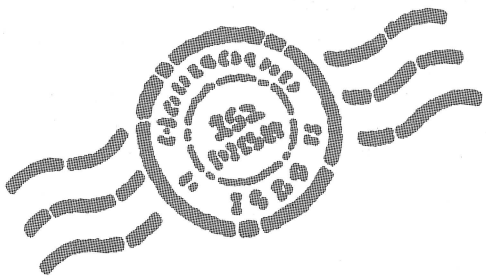
```
@performkeys("{Uplevel}")
```

*From then on (once you've saved the library), **Alt-U** will perform the **Uplevel** for you.*

Thanks for the reply. It does indeed work. I have found a better solution, though. On this particular computer (Zenith 181), the key combination **Fn** (a special key) and **F1** generates **F11** and **Fn-F2** generates **F12.** These two keys are defined in the standard IBM keyboard also as **Uplevel** and **Dnlevel**.

*Also, to make the numeric keypad (which is overlaid on some of the regular keys) work properly with Framework II, it is necessary to put the keyboard in "dumb" mode, as opposed to the default "smart" mode. "Dumb" mode is enabled with **Fn-2** and "smart" mode with **Fn-1**.*

# LETTERS



dBASE III PLUS

## DBSYSTEM.DB and PROTECT

Dear ***TechNotes***: After setting up user profiles and access levels for some of my files with the PROTECT.EXE program supplied in the dBASE Administrator, I found that some of the files in the directory, including the .CRP files and the DBSYSTEM.DB, were accidentally erased. We reinstalled the Administrator and copied our .CRP file backups into the right subdirectories. Although we can start the Administrator and get to the dot prompt, the system does not let us access the encrypted files. Can you help us? — PB

*Dear PB: The access-level information to use these files is stored in the DBSYSTEM.DB file. If you backed up your .CRP files, but not your DBSYSTEM.DB, the system won't be able to access those files without the information in DBSYSTEM.DB. If you still have the .DBF source files of those .CRP files you can reencrypt the files using the PROTECT program.*

*A good way to avoid this problem is to make the DBSYSTEM.DB file HIDDEN, SYSTEM, and READ/ONLY. This will ensure that only the most competent users or the administrator know the file exists and how to delete it.*

*Keep in mind that as soon as you save your work session, PROTECT.EXE will encrypt the databases used during that session. Since each database's encryption is different, each time changes are made to the DBSYSTEM.DB security file using PROTECT, you should back up the databases used on that day, along with the DBSYSTEM.DB file.*

## ACCESS() and DOS Access

Dear ***TechNotes***: How can I prevent users that have different rights in a multiuser application from accessing certain selections (applications) in the system? Also, how can I give the users DOS access without giving them access to the dot prompt, in order to avoid unauthorized users from accessing the network environment? — LY

*Dear LY: A program like the following one can be used as a main menu for your system. It checks the access levels given through the PROTECT.EXE program to the various users, using the ACCESS() function. Note the use of the &doscomm macro to prevent users from directly accessing the operating system.*

*The LAPSE program prevents the display of messages from clearing before the user has had a chance to read them.*

```
** Lapse.PRG
Lapse = 0
DO WHILE Lapse <= Timelength
       Lapse = Lapse + 1
ENDDO
```

*Timelength is a memory variable set by the developer or the database administrator.*

```
* Program ...: MainMenu.PRG
* Author ....: Richard Augelli
* Date ......: December 1, 1987
* Version ...: dBASE III PLUS, version 1.1.
* SET ENVIRONMENT COMMANDS
* SET DEFAULT PATHS AND LOGICAL DRIVES
*
Choice = 0
DO WHILE .T.
   CLEAR
   @ 3,30 SAY "* * * M A I N     M E N U  * * *"
   @ 7,25 SAY "1. ACCOUNTS MENU"
   @ 8,25 SAY "2. CLIENTS MENU"
   @ 9,25 SAY "3. PERSONNEL MENU"
   @10,25 SAY "4. INVENTORY MENU"
   @11,25 SAY "5. UTILITIES MENU"
   @12,25 SAY "6. USE WORD PROCESSOR"
   @13,25 SAY "7. USE SPREADSHEET"
   @14,25 SAY "8. EXECUTE DOS COMMANDS"
   @15,25 SAY "0. EXIT"
   @17,25 SAY "ENTER SELECTION: " GET Choice
   DO CASE
   CASE Choice = 0
      SAVE TO ACCOUNT
      RETURN
   CASE Choice = 1 .AND. ACCESS() <= 3
      DO Accmenu
   CASE Choice = 2 .AND. ACCESS() = 1
      DO Clients
   CASE Choice = 3 .AND. ACCESS() < 5
      DO Personnel
   CASE Choice = 4 .AND. ACCESS() < 6
      DO Inventry
   CASE Choice = 5 .AND. ACCESS() < 4
      DO Util
   CASE Choice = 6
      RUN &Wordproc
      WAIT
   CASE Choice = 7
      RUN &Ssheet
      WAIT
   CASE Choice = 8
      @20,0
      ACCEPT "Enter DOS Command you want to execute ->" TO doscomm
      RUN &doscomm
      WAIT
   CASE (Choice = 0 .OR. Choice > 8)
      @20,0 SAY "NOT A VALID OPTION. PLEASE RETRY"
      DO Lapse
   OTHERWISE
      @20,0 SAY "YOU CANNOT ACCESS THOSE FILES. PLEASE RETRY"
      DO Lapse
   ENDCASE
ENDDO
```

FRAMEWORK II

## Summing Even Numbers in a Spreadsheet

Dear *TechNotes:* I have a spreadsheet with a region that contains numeric codes in it. I need to consider only the even numbers in the region for purposes of consolidation. How can I get Framework II to calculate the sum of all the even numbers in a region of a spreadsheet ? — RE

*Dear RE: The following FRED program, Addeven.FW2, uses the @mod function to accumulate all the even numbers in a region. For instance, the formula of a cell could be:*

*@Addeven (ss.A1:ss.B4)*

*To use Addeven, it must be on the desktop or in the library.*

```
;
; Addeven.FW2
;
; Usage: @Addeven(region)
;
@local(total, cell),
total := 0,
@while((cell := @get(@item1)) <> #NULL!,
       @if(@mod(cell, 2) = 0,
          total := total + cell
       ),
       @next(@item1),
),
@return(total)
```

## @getenv()

Dear *TechNotes:* I am writing an application in FRED that accesses files in different directories. I need to capture, in a local variable, the PATHs that were SET in DOS before starting Framework II. Can you help me? — YU

*Dear YU: The function @getenv("PATH") will return the DOS PATH previous to starting Framework II. The following example shows a line of code that will capture the DOS PATH and store it to a local variable called dospath.*

```
@local(dospath),
dospath := @getenv("PATH")
```

## Background Processing

Dear *TechNotes:* Is it possible to write an application in FRED that performs some tasks, such as telecommunications or printing in the background? Can you give me an idea of how this can be done? — RA

*Dear RA: The* Framework II Developer's Toolkit *contains a low-level interface that lets you call low-level machine code routines from within FRED application programs. The service routines explained in the* Framework II Developer's Toolkit *documentation allow Framework II to execute background tasks such as print spooling or telecommunications.*

*Calls to the Framework II service routines are made by setting up registers to indicate the desired function and the object to act on and then performing a far, indirect call through the Framework II service routine address that was set to location zero in the module. This is documented in the* Toolkit *itself.*

# Corrigenda

## October 1987, page 31 Detecting Network Redirection

The last line of the article was truncated. It reads:

> If no redirection is in effect or if an error

It should read:

> If no redirection is in effect or if an error occurs, the variable will be returned unchanged.

## October 1987, page 78 Calcsize.FW2

The last two lines of the program were omitted:

```
@eraseprompt,
@prompt ("Size is " & @integer(size))
```

# Cumulative Index
## January 1987 through December 1987

**Products:**

| | |
|---|---|
| D2 | dBASE II |
| DP | dBASE III PLUS |
| DPA | dBASE III PLUS ADMINISTRATOR |
| DB | dBASE Products |
| FW2 | Framework II |
| MM | Multimate |
| RF | RapidFile |
| UT | Utilities |


$ operator, DP Sep:30
@...GET, DP Apr:59, DP Dec:64
@...SAY, DP Apr:59, DP Apr:64, D2 May:67, DP Dec:64
  to text files, DP May:3, DP Jun:77, DP Aug: 45
  using drivers, DP May:12
  nondestructive cursor positioning, DP Jul:28
@...SAY...PICTURE
  with negative numbers, DP Jun:27
Abbreviations
  damaged, FW2 Sep:69
@abs, FW2 Mar:26


## A


ACCEPT TO, DP Sep:32
ACCESS() DPA Feb:50, DP Dec:67
Add_if.FW2, FW2 Jan:46, FW2 Jul:66
Adding record numbers, RF May:48
Adding users on a network, DPA Apr:12
Addlabel.FW2, FW2 Aug:49
ADDUSER.COM, DPA Apr:3, DPA Apr:12
Adjust.FW2, FW2 Aug:51
Adjusting field width, RF Jul:53
Alt-1.FW2, FW2 Jul:44
Alt-B.FW2, FW2 Jul:46
Alt-C.FW2, FW2 Jul:46, FW2 Aug:52
Alt-F.FW2, FW2 Jul:47
Alt-H.FW2 Dec:47
Alt-K.FW2, FW2 Jul:43, FW2 Aug:49
Alt-P. FW2 Dec:53
Alt-V.FW2, FW2 Jul:47
ALTERNATE = in CONFIG.DB, DP Jan:22
Alternate keys, DP Sep:36
@and(), FW2 May:66
APPEND, DP Jan:44, DP Mar:13, FW2 Mar:61, DP May:19, RF Aug:69, DPA Aug:73, DP Dec:64
  correctly updating indexes on a network, DPA Jul:22
  to a DELIMITED file, DP Jun:70
APPEND FROM, DP Jul:63, DP Sep:3
  DELIMITED, DP Aug:43
Apps:Mailmerge using, FW2 Sep:75
Arrays
  imitating in dBASE III PLUS, DP Jul:26
  save and restore, UT Mar:54
ASC(), DP Apr:16
ASSIST, DP Apr:35, DP Apr:41, DP Jun:27, DP Sep:33
AT(), DP Feb:63, DP Jul:14
ATO
  with CompuServe, FW2 Jun:69
AT&T 7300, DP Jul:69
ATTRIB, DPA Apr:15
attributes, FW2 Aug:53
AUTOEXEC.BAT, DP Jan:58
@avg, FW2 Feb:10


## B


Back.PRG, DP Sep:73
Background processing, FW2 Dec:69
BACKUP.EXE, DP Jul:3
@beep, FW2 Jan:59
BELL = in CONFIG.DB, DP JAN:22
BIOS problems, RF Dec:63
Break character, FW2 JUL:39
BROWSE, DP Jan:45, DP Sep:33, DP Sep:60, DP Sep:61
  EGA 43-line mode, DP Feb:56
  WIDTH, DP Feb:47
BUCKET = in CONFIG.DB, DP Jan:21


## C


Calculated fields, RF May:29, RF Jun:77
Calculations
  clearing results of, FW2 Dec:59
CALL, DP Jan:28, DP Feb:21, DP Mar:19, DP May:44, DP Apr:59
CANCEL, DP Aug:66, DP Aug:67
Capslock key, FW2 Aug:70
CARRY = in CONFIG.DB, DP Jan:22
Case folding, DP Jan:61
CATALOG = in CONFIG.DB, DP Jan:22
Catalogs, DP Jan:64, DP Jul:68
CENTURY = in CONFIG.DB, DP Jan:22
CHANGE, DP Apr:59
Chdir.ASM, DP Sep:21
Chgtype.PRG, DP Aug:29
Chkfiles.PRG, DP Aug:26
Chkstat.PRG, DP Aug:38
CHMOD, UT Apr:15
@chr(), FW2 Mar:25
CLEAR, DP Sep:33
CLEAR ALL, DP Aug:66
CLOSE PROCEDURE, DP Apr:40
Collation sequence, DP Jan:61
COLOR = in CONFIG.DB, DP Jan:22
Color
  blinking, DP Jun:28
  high-intensity, DP Jun:28
Columnar listings, DP Aug:41
COMMAND = in CONFIG.DB, DP Jan:21, DP Apr:35
Commas
  in formatted cells, RF Aug:60
Comments, DP Jul:62, FW2 Dec:59
Compaq 386, DP Feb:47
  DOS, UT Apr:38
Compaq III Portable

screen driver, FW2 Sep:56
Compound interest
calculating, FW2 Dec:60
CompuServe
downloading articles from, Aug:3
COMSPEC, UT Apr:39, DP May:59
CONFI256.SYS, DP Feb:64
CONFIG.DB, DP Feb:47, DP Feb:64, DP Jan:70, DP Apr:35, DPA Jul:20, DP Sep:34
classes of commands, DP Jan:20
configuration commands, DP Jan:21
creating and editing, DP Jan:20
function keys, DP Jan:25
SET commands, DP Jan:22
setting colors, DP Jul:28
CONFIG.SYS, DP Jan:40, DP Feb:64, UT Apr:40, DP Jun:29, DP Aug:70, DP Sep:61
BUFFERS, DP Jan:19, DPA Aug:6
FILES, DP Jan:19
CONFIRM = in CONFIG.DB, DP Jan:22
CONSOLE = in CONFIG.DB, DP Jan:22
CONTINUE, DP Apr:41
Control key shortcuts, RF Sep:57
Converting
numbers to words, DP Mar:17
COPY, FW2 Jan:9, DP Jan:45, FW2 Mar:62, FW2 Apr:22, FW2 Apr:60, DP Dec:65, DP Sep:3, DP Sep:61
absolute cell references, FW2 Mar:29
boldface text, FW2 Mar:29
frames, FW2 Mar:29
soft hyphens and page breaks, FW2 Mar:30
STRUCTURE EXTENDED, DP Mar:52
using the FIELDS phrase, DP May:22
COPY TO, DP Dec:38
reclaiming disk space, DP May:20
COUNT, DP Dec:65
Cpsep.PRG, DP Jan:45
CREATE, DP Mar:52
memo fields, DP May:19
CREATE LABEL, DP Feb:48
CREATE REPORT, DP May:45, DP Dec:37
CREATE SCREEN, DP Dec:61, DP Mar:10
CREATE VIEW FROM ENVIRONMENT, DP Dec:61
Ctrl-C
pressing while printing, RF Sep:65
Ctrl-F10, FW2 Jul:37
Ctrl-Return, FW2 Jan:9
Ctrl-S, FW2 Jul:3
Current system date, DP Jun:70
Cursor pointing, FW2 Jun:42
Cursor positioning
nondestructive, DP Jul:28
Custom reports, RF May:48
Cut and Paste, FW2 May:61, FW2 Jul:46, FW2 Aug:53


# D


Data entry, FW2 Feb:10
field ditto, DP Mar:13
validation, RF May:30
Data Terminal Ready (DTR) signal, FW2 Jul:36
Database records
selection of, DP Sep:14
DATE(), DP Jan:62
Date calculations, DP Dec:35
Date fields, DP Dec:61
Date formats, DP Sep:76
Dates
conversions, DP Mar:61
display BC, RF Aug:60
display from other centuries, RF Aug:60
exporting to dBASE, FW2 Dec:66
importing from dBASE III PLUS, FW2 Aug:57
logical comparisons, DP Jul:24
manipulating for business, DP Dec:35
test for blanks, DP Mar:69
using in match criteria, RF Jul:54
DATES.ASM, FW2 Jun:35
Day.C, FW2 Jun:35
Day.FW2, FW2 Jun:37
dBASE Administrator, DPA Dec:22
correctly updating indexes with APPEND, DPA Jul:22
creating a shared directory, DPA Apr:5
DISPLAY STATUS, DPA Apr:57
DISPLAY USERS, DPA Apr:34, DPA Apr:57
EDIT, DP Jul:22
files to erase when reinstalling, DPA Aug:46
IBM problems, DPA Apr:34
modifying network parameters, DPA Aug:8
Novell network MAP command, DPA Aug:45
problems on 3COM 3+, version 1.1, DPA Aug:45
protect database re-encryption, DPA Apr:34
running on a 3COM 3+ network, DPA May:57
running from a workstation, DPA Apr:14
dBASE III PLUS LAN PACK
Novell network, DPA Aug:73
@dbasefilter, FW2 Feb:58, FW2 May:61, FW2 Aug:53
dBASE import, FW2 Dec:66
dBASE Programmer's Utilities, DP Jul:65
dBASE TOOLS: The Graphics Library for C, DP Sep:69
DBC, DP May:58
dbCODE, DP Aug:44, DP Aug:66, DP Dec:62
.DBF file, DP Jul:7
format, DP Dec:65
to locate EOF markers embedded in files, DP Jul:7
to recover from a DOS BACKUP without RESTORE, DP Jul:7
DBF header structure, DP Jul:6
DBF(), DP Jan:64
DBL, DP May:58
DBNETCTL.300, DPA Apr:3, DPA Apr:13
DBSYSTEM.DB, DPA Apr:15
DCONVERT, DP Jan:57, DP Apr:58, DP Sep:62, DP Dec:37
DEBUG = in CONFIG.DB, DP Jan:23
DEBUG.COM
general syntax, DP Jul:3
internal registers, DP Jul:5
memory addresses, DP Jul:4
to change the first byte of a .DBF file, DP Jul:7
to locate EOF markers embedded in files, DP Jul:7
to recover from a DOS BACKUP without RESTORE, DP Jul:7
DECIMALS = in CONFIG.DB, DP Jan:23
DEFAULT = in CONFIG.DB, DP Jan:23, DP Mar:9
DELETE, DP May:20
DELETED = in CONFIG.DB, DP Jan:23
DELETED(), DP Jul:63
DELIMITED file
APPEND TO, DP Jun:70
DELIMITERS = in CONFIG.DB, DP Jan:23
Design layout, RF Jul:54
DEVICE = in CONFIG.DB, DP Jan:23
@diffdate, FW2 Apr:31
DIR, DP Mar:9, DP Aug:44
Directories
changing from dBASE III PLUS, DP Sep:19
printing, RF Sep:65
DIRECTORY, DP Aug:44
@display, FW2 Jan:59
DISPLAY, DP Jun:29
Display Labels, FW2 Sep:70
DISPLAY STATUS, DP Mar:63, DPA Apr:57, DPA Jul:20, DP Sep:19
DISPLAY USERS, DPA Apr:15, DPA Apr:34, DPA Apr:57
Disptest.PRG, DP Jun:17
Ditto.PRG, DP Mar:14
DO, DP Sep:34
DO CASE
using arrays to replace, DP Jul:27
Do Consolidate.FW2, FW2 May:27
DO...WITH, DP Jul:24
DOS
installing on the system disk, RF Sep:66
DOS access, FW2 Jan:15, DP Dec:67
prompt change, FW2 Apr:32
DOS messages
burying, DP Dec:65
DRAG, FW2 Mar:31
Drive cabinets, FW2 Jun:43
Drive settings, FW2 Jan:8
Drives
retrieving from dBASE III PLUS, DP Sep:19
DTOC(), DP May:60

Dual column macro, FW2 Apr:23
*See also:* Printing with Framework II

# E

ECHO = in CONFIG.DB, DP Jan:23
EColor43, DP Sep:62
EDIT, DP Apr:59
memo fields, DP May 20
minimizing locked time on a network, DPA Jul:22
Editing
display hidden characters, FW2 Mar:31
documents, FW2 Jan:46
Undo, FW2 Mar:31
Editrec.PRG, DP Sep:15
EDIX, DP Jan:36
EJECT, D2 May:67
EGA
43-line mode, DP Jun:15
Empty.PRG, DP Feb:28
*See also:* Word wrapping
Enable color, RF Sep:58
End of Line Char Count, FW2 Jul:42
End-of-file markers
using DEBUG to repair, DP Jul:3
ENDSPOOL
using on the Novell Advanced NetWare network, DPA Jul:20
EOF(), DP Apr:38
ERASE, DP Dec:62
field, DP Jun:73
Error messages
trapping, DP Sep:35
Error messages in Assembler
"Warning: no stack segment", DP Jul:16
Error messages in dBASE III PLUS
"***Execution error on +", DP Feb:26
"***Execution error on SUBSTR():...", DP Apr:40
"***INTERRUPTED***", DP Aug:66
".DBT file cannot be opened", DP Feb:48, DP May:24
"ALIAS name already in use", DP Aug:44
"Alias not found", DP Feb:57, DP Mar:63
"Cannot erase a file which is open", DP Jul:25
"CONTINUE without LOCATE", DP Apr:41
"Do you wish to delete field", DP Mar:10
"Execution error on LOG(): Zero or negative", DP May:45
"File does not exist", DP Apr:42
"File is already open", DP Apr:42
"File not found", DP Jun:73
"Illegal value", DP Feb:47
"Internal Error Illegal OP Code", DP Jul:64
"Invalid DIF vector - DBF field mismatch", DP Apr:38
"Invalid printer port", DP Apr:41
"NET 803: Network path not found", DPA Mar:10
"Not a character expression", DP May:44
"Not a dBASE database", DP Jul:8
"not readable", DP Mar:9
"Numeric overflow (data was lost)", DP Feb:49
"Printer is either not connected or turned off", DP Apr:41
"Printer not ready", DP Mar:11, DP Apr:41
"Syntax error in field expression", DP Jan:57, DP May:45
"Syntax error", DP Mar:63, DP Jul:63
"Table is full", DP Oct:38
"Variable not found", DP Jan:45, DP Mar:63
Error messages in Framework II
"Frame badly formatted - unable to read it", FW2 Mar:25
"Missing operand", FW2 Aug:53
"Not able to load extended memory driver", FW2 Aug:53
"Not enough normal memory to keep overlays...", FW2 Aug:70
"Not enough normal memory to use extended memory", FW2 Jan:11
"Read error while trying to synch disk", FW2 Jul:40
"Telecomm now inactive", FW2 Jul:37
"Waiting for trap character", FW2 Jul:38
Error messages in RapidFile
"Current memo too large...", RF Jun:51
"File <filename> not found", RF Jul:54
"Invalid range", RF May:30
ERROR(), DP Sep:35, DP Dec:37
ESCAPE = in CONFIG.DB, DP Jan:23
European versions, DP Jan:60
EXACT = in CONFIG.DB, DP Jan:23
EXP(), DP May:45
Export.PRG, DP Sep:11
Exporting
@...SAY to a text file, DP May:3
ASCII file, DP Jan:45
from layouts, RF May:48
memo fields, DP May:21
of database files, DP Sep:3
saving data to a text file, FW2 Apr:56
SYLK files, FW2 Mar:38
text files, DP May:22
Expressions
nested, RF Aug:60
EXTEND SELECT, FW2 Jan:46, FW2 Jan:58
Extended memory, FW2 Jan:10, FW2 Aug:70, FW2 Dec:58
on the Compaq Portable, FW2 Sep:75
External programs
interfacing dBASE III PLUS, DP Mar:70

# F

Fields
adding to common numeric, DP Aug:72
changing names, RF Jun:50
totaling, RF Jun:51
Fields
select to list by numbers, DP Sep:31
FILE(), DP Sep:62, DP Dec:37, DPA May:58
File corruption, DP Aug:70
File format, DP Dec:65
File handles, FW2 Mar:30
File to load on startup, RF Jun:50
File structure table, RF May:30
File transfers, FW2 Jul:41
Files
necessary, FW2 Sep:70
saving, RF Jun:51
Fill.PRG, DP Mar:54
Fillcol.FW2, FW2 Aug:50
Filtering, FW2 Mar:71
FIND, DP Sep:14
Fkeydef.PRG, DP Dec:8
Fkeyset.PRG, DP Dec:12
Floating dollar signs, DP Apr:39
REPORT FORM, DP Aug:41
FOR, DP Apr:41
Formal definition of a keyword, FW2 Apr:31
Form letters
blank fields, RF May:48
Forms view, FW2 Mar:32
Frame Label
printing defaults, FW2 Sep:70
Framelock, FW2 Jan:63
Frames
creating sized, FW2 Jun:43
forcing open or closed, FW2 Dec:59
size, FW2 Jan:12
Framework II Developer's Toolkit, FW2 Dec:69
Numlock and Capslock keys, FW2 Aug:70
using with Microsoft C, FW2 Jun:30
FRED programs, FW2 Jun:39
FUNCTION = in CONFIG.DB, DP Jan:25
Function keys, DP Dec:4
reprogramming to recognize DOS functions, DP Apr:64
Functions
comparing dBASE III PLUS and RapidFile, RF Aug:54
using in match criteria, RF Jul:53

# G

Genie.PRG, DP Sep:8
Getdrive.ASM, DP Sep:22
GETENV(), DP Apr:39, DP May:59
@getenv(), FW2 Dec:69
Getkey.ASM, DP Dec:41
GETS = in CONFIG.DB, DP Jan:21
Getstat.PRG, DP Aug:35
Graphics library
save and restore arrays, UT Mar:54
scaling, UT Mar:55
Graphing, FW2 Mar:32, DP Sep:69
GraphLink

graphics integration, MM Jan:41

## H

Hard disk
crash and copy protection, DP Sep:72
files backing up, DP Sep:73
Hayes modem, FW2 Dec:62
Headers
in Quick Reports, RF Jun:50
HEADINGS = in CONFIG.DB, DP Jan:23
HELP, FW2 Jan:3, FW2 Mar:32, DP Sep:35
HELP.DBS, DP May:45
HELP = in CONFIG.DB, DP Jan:23
@hide, FW2 Mar:31
HISTORY = in CONFIG.DB, DP Jan:23
HP LaserJet, MM Apr:26, MM Apr:43, FW2 May:69, MM Jun:75
line and box drawing with, MM Jun:52
printing to, RF Jun:50
soft fonts, MM Jun:75
*See also:* Printing with MultiMate
HP-3000, DP Jul:69
Hypertext, FW2, Dec:44

## I

IBM PC Network Program
fine-tuning, DPA Aug:5
IBM Token-Ring Network with MultiMate, MM Jun:75
Icaps.PRG, DP Dec:40
ID, FW2 Jan:4
ID.EXE, DP Sep:36
IDLAN, DPA Apr:10
IFSTR$(), DP May:30
IIF(), DP Apr:40, DP May:60, DP Jul:64
Import.PRG, DP Sep:13
Importing
ASCII, MM Jun:58
comma separated text file, FW2 Apr:60
dBASE II/III FILES, FW2 MAR:30
DELIMITED, DP Jan:64
external text files, DP May:22
file capture in emulation, FW2 Jan:47
from a mainframe, DP Jan:64
Lotus 1-2-3 files into Javelin, UT Feb:59
memo fields, DP May:21
SDF, DP Jan:64
SYLK files, FW2 Mar:38
WordStar, FW2 Mar:31
INDEX, DP Jan:44, DP Feb:48, DP Mar:9, DPA Mar:10, DP Mar:11, DP Apr:58, DP May:59, DP Jul:64, DP Aug:67
Indexing, DP Jan:62
correctly updating with APPEND, DPA Jul:22
file size, DP Jan:44
key expression of, DP Dec:39
numeric fields, DP Apr:58
INKEY(), DP Feb:48, DP Apr:16, DP Apr:59, DP Aug:67, DP Sep:36, DP Sep:63
Inkey.ASM, DP Apr:19
Init Modem String, FW2 Sep:70
Initial caps
converting to, DP Dec:38
@inputline, FW2 Mar:61, FW2 Aug:53
INSTALL.BAT, DP Aug:44
Installation, FW2 Jan:4, FW2 Jan:9
creating a shared dBASE Administrator directory, DPA Apr:11
creating shared database and program directories, DPA Apr:6, DPA Apr:12
custom drivers, FW2 Jan:16
dBASE Access, DPA Apr:7
dBASE Administrator, DPA Apr:6, DPA Apr:9, DPA Apr:11, DPA Jun:27
dBASE III PLUS on the Starlan, DPA Apr:3
files required to run dBASE III PLUS, DP Aug:44
@int(), FW2 Mar:26
INTENSITY = in CONFIG.DB, DP Jan:23
Interest rate, FW2 Sep:53
International characters
sorting, DP Sep:74

## J

Javelin, UT Feb:59
JOIN
databases that contain memo fields, DP Aug:18

## K

Key macros, RF Jun:45
Keyboard drivers, FW2 Jan:8
for Tandy 1200 HD, FW2 Mar:32
Keyboard LEDs, RF Jul:53

## L

LABEL FORM, DP Jan:56, DP Jun:20
multiple labels, DP Jul:69
TO FILE, DP Dec:62
using drivers, DP May:12
Label Format
blank lines in, DP Dec:37
Label printing, FW2 Sep:38
Labels, FW2 Mar:27
LASTDRIVE = in CONFIG.SYS, DP Jan:40
Layout:Design Layout, RF Dec:63
Layouts, RF May:49
saving, RF Aug:61
Leading Edge Word Processor with dBASE III PLUS, DP Jun:72
LEFT$(), RF Jul:53
LEFT(), DP Jul:14
Leftarrow key
trapping it with Inkey.BIN, DP Apr:16
Library
macros, FW2 Mar:32
Library.FW2, FW2 Mar:33
Lines.ASM, DP Jun:15
LIST, DP Sep:31
ALL, DP Feb:63
one page at a time with headings, DP Jul:70
TO PRINT, DP Mar:11
LIST PROCEDURE, DP Apr:40
List.PRG, DP Jul:69
ListNpik.PRG, DP Sep:16
LOAD, DP Jan:28, DP Feb:21, DP Mar:18, DP Apr:59, DP Jun:4
LOCATE, DP Apr:41
Locate, FW2 Mar:33
Locate:Replace
capitalization, FW2 Sep:55
Logical fields and variables, DP Jan:62
Logical operators, DP Dec:37
LOG(), DP May:45
LOGIN.DB, DPA Apr:34
Lotus 1-2-3, FW2 Feb:10
converting files from, DP Aug:68
importing from, DP Sep:71
importing into Framework II, FW2 Jun:73
*See also:* Range input
Lstbynum.PRG, DP Sep:31

## M

Mac.INC, DP Jan:28
Macros, DP Jan:36, DP Jan:58
append to cut paste buffer, FW2 Jul:46
auto indentation, FW2 Mar:55
convert spreadsheet cell to numeric constant, FW2 Jul:47
database entry keyfilter, FW2 Jul:44
dual column, FW2 Apr:21
extra storage of, FW2 Oct:62
global, RF Oct:64
key, RF Jun:45
remove blank lines from word frames, FW2 Jul:45
scroll rate, FW2 Apr:34, FW2 Jun:76
wildcard directory information, FW2 Jul:46
Mailing labels, MM May:34
Mailmerge, FW2 Mar:27, FW2 Sep:38
from multiple databases, FW2 Jul:30
Mailmerging
one record, FW2 Sep:75
MainMenu.PRG, DP Dec:68
Make.PRG, DP Mar:53
Makescr.PRG, DP Jun:16
Makesdf.BAS, DP Nov:34
Makesdf.C, DP Nov:30
Makesdf.PAS, DP Nov:32
Makestru.PRG, DP Aug:27
MARGIN = in CONFIG.DB, DP Jan:23
Match criteria
using dates, RF Jul:54
Matching records, RF Dec:63
Math co-processor, RF Oct:64
Matrix multiplication, FW2 Oct:47
MAXMEM = in CONFIG.DB, DP Feb:47
Mem2char.ASM, DP Aug:29

Memo fields, DP Feb:48, DP Apr:59
accessing from edit commands, DP May:20
converting, DP Nov:15
editor, DP May:21
joining databases that contain, DP Aug:18
reporting, DP May:22
searching and substringing, DP May:23
Memo2dbf.PRG, DP Nov:18
Memochar.PRG, DP Nov:20
Memojoin.PRG, DP Aug:24
Memoline.PRG, DP Nov:23
Memory, FW2 Jan:9, FW2 Nov:65
extended, FW2 May:69, FW2 Dec:58
loss, FW2 Nov:65
optimization, RF May:49
overlays, FW2 Jan:10
virtual, RF May:49
Memory addresses, DP Jul:4
*See also:* DEBUG.COM
Memory variables, DP Oct:41
PUBLIC vs. PRIVATE, DP Jul:63
Memos
personalizing, RF Dec:63
Memotype.PRG, DP Nov:21
MEMOWIDTH = in CONFIG.DB, DP Jan:23, DP Feb:47
@menu, FW2 Jan:58
MENU = in CONFIG.DB, DP Jan:23
MENUBAR.BIN, DP Oct:38
Merge fields, RF Oct:64
Merge printing
multiple databases, FW2 Jul:30
Microsoft C Compiler
using with the Developer's Toolkit, FW2 Jun:30
MID$(), RF Jul:53
@mid(), FW2 Mar:26
MOD(), DPA Jul:25, DP Nov:67
@mod, FW2 Dec:69
MODE, FW2 Jan:8, DP Feb:47, UT Apr:32, UT Apr:38
Modems
telecommunications questions and answers, FW2 Jul:36
MODIFY
memo fields, DP May:19
MODIFY COMMAND, DP Jan:36, DP Jan:57, DP Jan:62, DPA Jul:25, DP Nov:67
international characters, DP Aug:43
*See also:* Program editors
MODIFY FILE, DP Nov:67
MODIFY REPORT, DP Jan:57
MODIFY SCREEN, DP Mar:10, DPA Jul:25
MODIFY STRUCTURE, DPA Mar:10, DP May:24, DPA Jul:25
reclaiming disk space, DP May:20
MODIFY VIEW, DP Jul:26
Monitors
dual, FW2 Sep:69
Month.FW2, FW2 Jun:38
MontoNum.FW2, FW2 Mar:61
Mouse, FW2 Apr:60
Mouse driver, DP Dec:64
MOVE, FW2 Jan:9
Multimate
export from, FW2 Oct:69
Multimerge.FW2, FW2 Jul:32
Multiple labels, DP Sep:68
MVARSIZ = in CONFIG.DB, DP Jan:21
Mymain.ASM, FW2 Jun:32


# N


#NAT, FW2 Nov:65
@nationalize, FW2 Sep:60
Ndxsize.PRG, DP Jan:44
Negative numbers, RF Jun:51
used as length arguments, RF Jul:53
NETBIOS, DP OCT:71
Network redirection
corrigendum, DP Dec:69
detecting , DP Oct:30
Newsave.ASM, DP Jun:6
*See also:* Screens
@nextkey, FW2 Jan:59
Norton Editor, DP Jan:37
@not(), FW2 May:66
NOT, DP Oct:7
Novell network, DP Jan:46
installing dBASE III PLUS on, DP Sep:23
MAP command, DPA Aug:45
merge printing documents on, MM Jun:75
SPOOL, DPA Jul:20
NUM(), RF May:30
Num2Word.ASM, DP Mar:19, DP Sep:66
Numbers
summing, FW2 Dec:69
Numeric coprocessor, FW2 Jan:8
NumLock
toggle, DP Oct:5


# O


ON ERROR, DPA Jul:25, DP Sep:35, DP Dec:62
ON ESCAPE, DP Mar:63, DP Aug:66, DP Dec:62
ON KEY, DP Dec:62
@Openframe, FW2 Dec:59
@or(), FW2 May:66
.OR. operator, DP Sep:30, DP Oct:7
XOR, DP Aug:40
Outline view, FW2 Mar:34
Overlays, FW2 Jan:10


# P


PACK, DP May:20
@panel, FW2 Mar:62, FW2 Jun:43
PARAMETER
misspelling keyword, DP Jul:24
Parameters
passing, DP Feb:22, DP Mar:18
Parens.C, FW2 Dec:55
Parentheses
matching, FW2 51
PATH, UT Apr:39, May:59
PATH = in CONFIG.DB, DP Jan:24
Path length limitations, DP Sep:37
PEEK(), D2 Apr:59
@performkeys, FW2 Jan:58, FW2 Mar:61, FW2 Jun:39
Personal use extension, DP Sep:68
PFS files
importing, RF Sep:65, DP Oct:38
PFS:FILE attachments, RF Sep:58
Pmain.ASM, FW2 Dec:55
Point & dial, FW2 Jul:36
POKE, D2 Apr:59
Port assignment, FW2 Jan:4
Pos_Neg, FW2 Oct:77
PRINT = in CONFIG.DB, DP Jan:24
Printer
control, FW2 Jan:11
driver, FW2 Jan:4, DP May:10, FW2 Oct:74
timeout, FW2 Sep:56
Printer Action Tables, MM Feb:3, MM Feb:58, MM Apr:26, MM Apr:42, MM May:50, MM May:63
Printing with dBASE III PLUS, DP Apr:41, DP Dec:65
compressed mode, DP Jul:64
configuring drivers, DP May:10
field names as headers, DP Jun:72
logical and character fields combined, DP Aug:67
program listings, DP Nov:75
sending CHR(0), DP May:13
using European versions, DP Jan:61
@Z, DP Jan:56
Printing with Framework II
a database in dBASE view, FW2 Jan:63
bolding frame labels, FW2 Jun:74
British pound symbol, FW2 May:67
driver for NEC 7710, FW2 Mar:35
dual column, FW2 Apr:21
elite type on the Okidata 193, DP Jul:71
footers, FW2 Feb:50
format/control, FW2 Mar:35
frame labels, FW2 Mar:35
from virtual memory, FW2 Mar:35
graphic characters, FW2 May:70
headers, FW2 Feb:50
maximum lines stranded, FW2 Jun:73
orienting spreadsheets with SIDEWAYS, FW2 Apr:34
page breaks, FW2 Jul:66
removing /p, FW2 Jun:73
row and column labels in a spreadsheet, FW2 Aug:47
selected records from a database, FW2 Mar:71
translating foreign languages, FW2 Jun:41
underlining headers, FW2 Jun:74
using attributes, FW2 Mar:56
Printing with Multimate

attributes, MM Apr:26
creating endnotes, MM Mar:48
from dBASE files, MM Jun:59
legal-size and letter-size documents, MM Apr:42
logic, MM Feb:3
mailing labels, MM May:34
math font cartridges, MM Apr:27
merging, MM Mar:57
missing characters, MM Apr:43
pitch table, MM Apr:28
Printer Action Tables, MM Feb:3, MM Feb:58, MM Apr:26, MM Apr:42
settings for the HP LaserJet, MM May:50
using the HP LaserJet, MM Apr:26
using the WPS format, MM Mar:48
Printing with RapidFile
aborting print operations, RF Sep:58
codes, RF May:49
settings, RF May:49
using calculated fields, RF May:29
Privileges, DP Jan:45
PROFS, FW2 Jul:66
Program design
modularity, DP Feb:31
Program editors, DP Jan:35
WordStar, DP Jan:57
Programming with dBASE III PLUS using European versions, DP Jan:62
Programs, DP Sep:36
Projtime.PRG, DP Oct:37
PROMPT.= in batch files, UT Apr:32
PROMPT = in CONFIG.DB, DP Jan:21
PROTECT, DPA Mar:10, DPA May:58, DP Dec:22, DP Dec:67
Prtlab.PRG, DP Jul:69


# Q


Quick Report, RF Oct:70


# R


Random Access Memory (RAM) disk, FW2 Jan:10
Range input, FW2 Feb:10
Range.FW2, FW2 Feb:14
*See also:* Range input
RangeInput.FW2, FW2 Feb:15
RapidFile
comparing dBASE III PLUS and RapidFile functions, RF Aug:54
@rate, FW2 Sep:53
READ, DP Apr:59
Read-only files, RF Jul:54
READKEY(), DP Oct:75, DP Sep:30
Record tagging, FW2 Nov:76, FW2 Sep:53
RECSIZE(), DP Oct:67
Recursive calls, DP Sep:34
Referencing
library, FW2 Mar:36
regions, FW2 Mar:36
with English names, FW2 Mar:36
Region-walking, FW2 Oct:45
Regions
pseudo-random access to, FW2 Oct:45
REINDEX, DP Jan:44, DP Jul:64
RELEASE ALL, DP Jul:25
RENAME, DP Sep:37
REPLACE, DP Feb:49, DPA Mar:10, DP Jun:70, DP Jun:71, DP Aug:72
ALL, FW2 Mar:61, DP Oct:39
one character with another within a string, DP Jul:14
one string with another within a field, DP Aug:42
Replace.ASM, DP Jul:17
Replace1.PRG, DP Jul:16
Replace2.PRG, DP Jul:17
REPORT FORM, DP Jan:56, DP Apr:59, DP Apr:64, DP May:59, DP Sep:63, DP Nov:39
floating dollar signs, DP Aug:41
memo fields, DP May:22, DP May:45
menu creations, DP Feb:30
retrieval specifications, DP Aug:68
snaked column reporting, DP Jun:20
Report statistics
handling using report fields, RF Jul:48
Reports
break-point, FW2 Sep:44
cross-tabbed, DP Oct:75
custom, RF Oct:70
free-form, FW2 Sep:42
multiple copies of, RF Sep:65
snaked-column, FW2 Sep:40
wrapped-column, DP Dec:13
RESTORE.EXE, DP Jul:3
Restscr.ASM, DP Jan:32
RESUME, DP Aug:67
RETRY, DP Dec:27
Retry.C, DP Dec:29
RIGHT$(), RF Jul:53
RIGHT(), DP Jul:14
Rounding
in databases, DP Oct:63
Rptfrm.PRG, DP Feb:38
@run(), FW2 Mar:26
RUN, DP Jan:36, DP Jan:57
RUN/!, DP Oct:40
Running totals, FW2 May:47


# S


SAFETY = in CONFIG.DB, DP Jan:24
Sample.PRG, DP Aug:38
Savescr.ASM, DP Jan:30
Saveto.FW2, FW2 Oct:76
Saving
files, FW2 Mar:37
Say2File.ASM, DP May:6
@scan(), FW2, May:65
Sci.PRG, DP Mar:51
Scientific notation, DP Feb:49, DP Mar:51
SCOREBOARD = in CONFIG.DB, DP Jan:24
Screen drivers
Compaq III Portable, FW2 Sep:56
Screens
AT&T PC 6300, FW2 Jan:18
avoiding snow on the CGA, DP Jun:6
color, FW2 Jan:9
driver, FW2 Jan:4
Hercules Graphics Card, FW2 Jul:66
Hercules InColor Card, FW2 Jul:66
high-resolution, FW2 Jul:65
IBM Color Graphics, FW2 Jan:18
MDS Genius, FW2 Jul:66
partial save and restore to files, DP Jun:3
saving and restoring to files, DP Jan:26
Sigma, FW2 Jul:66
STB 400, FW2 Jul:66
Wyse WY700, FW2 Apr:33
SDF, DP Feb:49, DP Feb:57, DP Apr:57
Search conditions
multiple, DP Oct:75
SEEK, DP Sep:14
SELECT, DP Dec:64
@sense, FW2 Jun:39, FW2 Aug:52
Serial number protection, FW2 Jan:15
Serial port, DP Jul:65
Serial.PRG, DP Jun:71
SET
ALTERNATE, DP Jan:45, DP May:22, DPA May:58, DP Sep:3
CARRY, DP Mar:13
CATALOG, DP Jan:64, DP Mar:52, DP Jun:28, DP Jul:68, DP Aug:44, DP Nov:68, DP Oct:67, DP Dec:38
changes made by the applications generator, DP May:44
COLOR, DP Jun:28, DP Jul:28, DP Oct:71, DP Nov:39, DP Nov:68
CONSOLE, DP Dec:65
DATE, DP Jan:24
DELIMITERS, DP Jan:21, DP Dec:62
DEVICE TO PRINT, DP Mar:12, DP May:4
DEVICE TO SCREEN, DP May:4
DO HISTORY, DP Jan:24
ENCRYPT, DPA Jul:20
ESCAPE, DP Apr:16, DPA Jul:20, DP Nov:39
EXCLUSIVE, DP Mar:9, DPA Mar:10, DPA Jul:20
FIELDS, DP Jan:24, DP May:45, DP Oct:72
FILTER, DP Jan:24, DP Jun:29, DP Jul:63
FIXED, DP Jan:24
FORMAT, DP Jan:24, DP Oct:67, DP Nov:68
FUNCTION, DP Dec:4
HISTORY TO, DP Apr:64, DP Sep:37
INDEX, DP Jan:24, DPA May:58, DP Jun:28
MEMOWIDTH, DP Apr:59, DP May:22
MESSAGE, DP Jan:24
ODOMETER, DP Jan:24, DP Oct:67
ORDER, DP Jan:24
PATH, DP Jan:57, DP Apr:39, DP Jul:25, DP Sep:37
PRINT, DP Jan:24, DP Mar:12, DP May:22
PRINTER, DP Jan:24, DP Mar:12, DP

May:3, DPA Jul:22, DP Sep:63
PROCEDURE, DP Jan:24, DP Apr:42, DP Jun:22, DP Nov:69
RELATION, DP Jan:24, DP May:59, DP Aug:72, DP Sep:68, DP Oct:40
saving and restoring environment, DP Aug:35
SCOREBOARD, DP Aug:69
STATUS, DP Jan:58, DP Aug:69, DP Sep:34
TITLE, DP Jan:24
UNIQUE, DP Mar:11
@setselection, FW2 Feb:13, FW2 Jun:41, FW2 Jun:42
Setup
changing configurations, FW2 Jan:8
for the first time, FW2 Jan:4
library of configurations, FW2 Jan:18
primary hardware, FW2 Jan:8
secondary hardware, FW2 Jan:8
SHELL = in CONFIG.SYS, UT Apr:40
Shift-Tab, FW2 Dec:62
Sidekick!, DP Jan:38
SIDEWAYS, FW2 Apr:34
signed numbers
indexing, DP Dec:35
SIZE, FW2 Jan:63, FW2 Mar:37
@Sizeto, FW2 Jun:43, FW2 Nov:48
Snake.PRG, DB Jun:23, DP Dec:15
Snakgen.PRG, DP Dec:19
Snakmake.PRG, DP Dec:18
Snapshots
saving, RF Aug:61
SORT, DP Mar:10, DP Mar:12
Source, The
downloading articles from, Aug:3
Spelling check
macros, FW2 Mar:28
personal dictionary, FW2 Mar:28
repeated words, FW2 Mar:28
SPOOL
on the Novell Advanced NetWare network, DPA Jul:20
to print to a local printer, DPA Jul:22
Spooling, FW2 Jan:10
SPORT.BIN, DP Jul:65
Spreadsheet consolidation, FW2 May:25
Spreadsheets
altering values, FW2 Oct:76
calculating widths, FW2 Oct:78
rotating, FW2 Oct:48
Starlan network
adding users, DPA Apr:12
file server requirements, DPA Apr:4
subtracting users, DPA Apr:13
workstation requirements, DPA Apr:5
STATUS = in CONFIG.DB, DP Jan:24
Status bar, DP Sep:33
STEP = in CONFIG.DB, DP Jan:24
STORE, DP Sep:64, DP Oct:67, DP Jun:29
STR(), DP Apr:58, DP May:60, DP Aug:67, DP Sep:30
Strings
centered, DP Nov:37
justified , DP Nov:37
STUFF(), DP Jul:14, DP Oct:68
Subdir.PRG, DP Apr:36
Subdirectories in ASSIST, DP Apr:35
SUBST, DP Jan:40
SUBSTR(), DP Feb:64, DP Jul:14
Substring search, DP Feb:63
Subtracting users on a network, DPA Apr:13
@sum, FW2 Feb:10, FW2 Aug:69
@sumdate, FW2 Apr:31
SUSPEND, DP Aug:66, DP Aug:67
Sylkexp.FW2, FW2 Mar:45
Sylkimp.FW2, FW2 Mar:41

# T

@T, FW2 Apr:60, FW2 Jul:39
Tagging records, FW2 Nov:76
TALK = in CONFIG.DB, DP Jan:24
@tan(), FW2 Mar:26
Tandy 1000
keyboard driver, FW2 Jul:42
Tandy printers, DP Oct:40
.TCM files, FW2 Mar:25
TEDIT = in CONFIG.DB, DP Jan:22, DP Jan:57
Telecomm frame
moving active, FW2 Dec:63
Telecomm.TCM, FW2 Jul:38
Telecommunications
auto line feeds, FW2 Jan:46
changing default settings, FW2 Jul:38
questions and answers, FW2 Jul:36
Terminal emulation, FW2 Mar:64, DP May:61, FW2 Jul:41
VT100, FW2 Jul:66
Terminate-and-stay-resident programs, DP Sep:36
Sidekick!, DP Jan:38
@textselection, FW2 Apr:22
TEXT...ENDTEXT, DP Sep:64
Time
calculating, DP Oct:35
TimeFrame, FW2 Jan:63
Title.PRG, DP Nov:38
@today, FW2 Apr:31
"Too many files are open", DP Nov:67
Toshiba 1100+
Dnlevel/Uplevel, FW2 Feb:65
Toshiba keyboards, FW2 Dec:66
TOTAL, DP May:60, DP Sep:65, DP Dec:65
@trace(), FW2 Mar:26
Transaction report printing, FW2 Sep:49
TRANSFORM(), DP Jan:56, DP May:46, DP Dec:38
Translation, foreign language FRED programs, FW2 Jun:39
TRIM, DP Aug:41
TSR programs, DP Sep:36
TYPE(), DP Jul:26
TYPEAHEAD = in CONFIG.DB, DP Jan:24

# U

Underlining
removing in long documents, MM Jun:58
Uninstall, FW2 Jan:9, DP Jun:29
dBASE Access, DPA Apr:9
dBASE Administrator, DPA Apr:15
UNIQUE = in CONFIG.DB, DP Jan:24
Upgrading
from version 1.0, FW2 Jan:5
USE, DP Nov:69
USE...ALIAS, DP Oct:68, DP Dec:38
UTIL, MM May:63

# V

#VALUE! Error Constant, FW2 Sep:59
@var, FW2 Feb:10
VIEW = in CONFIG.DB, DP Jan:24
VIEW, FW2 Jan:63

# W

WAIT TO, DP Sep:32
WANG PC, FW2 Nov:76
Weekdays.PRG, DP Dec:36
WhichPRN.PRG, DP Oct:31
WHILE, DP Apr:41
@while, FW2 Jan:15
Word wrapping, DP Feb:21
WordPerfect, DP Sep:5
Wordstar, DP Jan:39
export from, FW2 Oct:69
Wordwrap.ASM, DP Feb:24
Wordwrap.BIN, DP Feb:21
WP = in CONFIG.DB, DP Jan:22, DP May:21
Wraptest.PRG, DP Feb:27
@writetextfile, DP Mar:26
Wsetargv.C, DP Dec:32

# X

XOR, DP Oct:7, DP Nov:76

# Y

Year.FW2, FW2 Jun:38

# Z

ZAP, DP Mar:52
Zapcis
with CompuServe, FW2 Jun:69
Zenith keyboards, FW2 Dec:66
Zero
dividing by, RF Jun:50
to the zero power, RF Jun:51
Zeroes
printing leading, DP Nov.75
ZOOM, FW2 Jan:9, FW2 Jan:15, FW2 Mar:37

# The Data You Save May Be Your Own

The *dBASE File Recovery* Support Pack, from the Ashton-Tate Support Products Center, provides a set of powerful, fully documented utilities for recovering data from damaged dBASE files. Since the data recovery process has been automated, it will save you hours of manual work as well as countless headaches. No dBASE user should be without it.

*dBASE File Recovery* gives you the tools to rescue your data. The heart of the Support Pack is a utility program that will find and repair data that might otherwise be totally lost. As long as the data is still physically present on your disk, it can be recovered. Because the utility operates independently of the disk directory, even data from a dBASE file that was totally destroyed (such as when floppy diskettes have been improperly swapped) can often be recovered intact.

Types of file damage that *dBASE File Recovery* can recover from include:

- Data that is "shifted" or out of alignment with the dBASE file's structure
- Files with corruption in the data area
- Files with missing or corrupted dBASE file headers
- Files that no longer appear in a directory but which still exist in some form on a disk
- Erased or ZAPped dBASE files
- Data from a backup that you cannot restore.

In fact, the only limitation on recovery of data is that your hardware must be able to read the storage medium, whether it's a floppy diskette or a hard disk.

*dBASE File Recovery* takes you step by step through diagnosing the problem with your dBASE file and recovering your data. A thorough discussion of how dBASE file structures work and how they can become corrupted provides background information that will allow even novices to diagnose dBASE file corruption problems.

With the *dBASE File Recovery* Support Pack, you can turn a major disaster into a minor mishap. It's data insurance for your dBASE files.

Order now at the Introductory Price of $39.95!

(See order form on reverse.)

# SUPPORT PACKS ORDER FORM

| Item | Price | Quantity | Total |
|---|---|---|---|
| **Support Pack #1: dBASE File Recovery** | *$49.95 | X_____ | =_____ |
| | **$39.95 | X_____ | =_____ |
| | (CA residents sales tax) | X %6.5 | =_____ |
| | (CT residents sales tax) | X %7.5 | =_____ |
| | subtotal | | =_____ |
| Ship via: ☐ UPS ($3.00) ☐ UPS BLUE ($13.00) | | | |
| ☐ U.S. MAIL ($3.00) ☐ Federal Express ($13.00) | $_____ | X_____ | =_____ |

**TOTAL PAYMENT ENCLOSED** (including shipping and sales tax): =_____

Name_____

Company Name_____

Address_____

City/State/ZIP_____

Telephone Number_____

Payment method:

☐ Check ☐ Money order ☐ Purchase order

☐ VISA ☐ Mastercard

Acct# ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Expiration date ☐☐ ☐☐

Signature_____

☐ Please send me information about other support packs for these products:

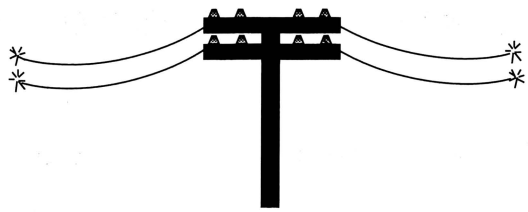
☐ dBASE ☐ LAN Pack ☐ Framework
☐ RapidFile ☐ dBASE Mac

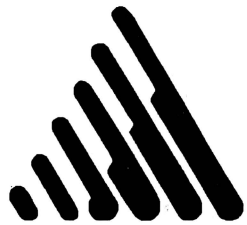
***orders received after December 31, 1987**
****orders received before January 1, 1988**

Mail this order form with your payment to:

**Ashton-Tate**
**Support Products Center**
**20101 Hamilton Avenue**
**Torrance, CA 90502-1319**
or call (213)-538-7482—Support Packs Only

A-T BBS





# ELECTRONIC SUPPORT

Your direct line to Ashton-Tate's technical archives begins with a modem and a call to our Support BBS. As a user of this service, you have access to more technical product information than you ever thought possible. Here are a few of the topics we offer:

1. Detailed **usage tips** to help you get the most out of each and every product feature. Things your manual never told you.
2. Descriptions of known **anomalies** ("bugs") and the most effective workarounds. *Knowing* is half the solution.
3. **Articles** which give step-by-step instructions for creating programs, macros, file systems, import/export, theory, and much more. Why reinvent the wheel?
4. New **drivers** for screens, printers, plotters, and keyboards. In minutes, you can download, install, and have running any of these late-breaking drivers. Who needs U.S. Mail?
5. **Sample files** and applications, including simple blocks of exemplary code, application utilities, DOS utilities, function libraries, multi-user programs, templates, and much more.
6. A busy **message board** where you can pose questions to Support Technicians or to other users. There are so many interesting ideas posted on the message board that you will learn something new every time you read it!
7. **Surveys** you can fill out to help us know what you want—now and for the future.
8. **Classified ads** to let other BBS members know about your own products and/or services.

A FREE service...24 hours a day!

**Electronic Support is the wave of the future... be a part of it!**

**(213) 538-6196** (modem)

# CLASSIFIEDS

All descriptions of services and products have been provided by the advertiser. These products and services have not been tested by Ashton-Tate and their inclusion in this section should not be taken as an endorsement of them by Ashton-Tate.

## SOFTWARE/ APPLICATIONS

**SOFSOLUTIONS CO.**
**ATTENTION: Genifer is now Quicksilver/ Clipper Compatible.** This powerful utility automatically converts any Genifer generated program into Quicksilver/Clipper compilable format. Just enter your file name and the program does the rest. Saves hours of debugging time! For "Genifer to Quicksilver" order GeniQuick. For "Genifer to Clipper" order GeniClip. Unlimited tech support. $59.00

**To order by phone call (512) 735-0746 or send a check to SofSolutions Co., 208 Senisa, San Antonio, TX 78228**

## USER GROUPS

dBASE SIG
St. Louis User's Group
for the IBM PC
P.O. Box 1345
St. Louis, MO 63043

dBASE Users Group
Picadilly Cafeteria
Centerville Rd and LBJ
Dallas, TX 73025

Modesto PC User Group
P.O. Box 5122
Modesto, CA 95352-5122

Space Coast PC Users Group
P.O. Box 396
Titusville, FL 32781-0396

**CONSULTANTS NEEDED**
**Accounting Software**

Are you familiar with dBASE III, do you understand small business accounting, and are you interested in providing consultant services under a strong profit sharing program on behalf of a six year old company with more than 10,000 installations nationwide? You will be responsible for generating new clients, recommending the appropriate accounting software, and providing ongoing client services. You can continue to operate your current business and be proud of the service you provide to your clients without compromising your "independent integrity." Recoup initial investment on first installation and start adding bottomline profits immediately. Call or write: Scott Robertson, Champion Business Systems, 17301 W. Colfax Ave., Golden, Co 80401.
1-800-243-2626 or 303- 278-8666.

Hawkeye PC User Group
P.O. Box 2966
Cedar Rapids, IA 52406-2966

University of South Florida
IBM PC Users Group
SVC 409
Tampa, FL 33620-7250

dBASE III Developers
Attn: Gil Orlik
2240 Buchanan, No. 5
San Francisco, CA 94115-9991

Philadelphia Area Computer
Society Framework SIG
2041 Harbour Drive
Palmyra, NJ 08065

Alaska Computer Society
6401 Downey Finch Dr.
Anchorage, AK 99516

San Diego Computer Society
IBM PC SIG
P.O. Box 87770
San Diego, CA 92138

Pemi-Baker Computer Users Group
RFD #2 Box 399
Plymouth, NH 03264

Richmond dBASE Users Group
4206 Seminary Avenue
Richmond, VA 23227

Tampa Database Users Group
119 Plantation Court, Suite D
Tampa, FL 33617-3731

The Boston Computer Society
One Center Plaza
Boston, MA 02108

Garden State Business Users Group
Lackawanna Plaza
Millburn, NJ 07041

Kansas City IBM PC Users Group
221 E. 3rd Street
Kansas City, MO 64113

UCLA PC Users Group
NPI 77-365, 760 Westwood Plaza
Los Angeles, CA 90024

Utah Blue Chips
P.O. Box 510811
Salt Lake City, UT 84151

NYPC dBASE SIG
40 Wall Street, Suite 2124
New York, NY 10005

Terre Haute PC User Group
100 South 22nd Street
Terre Haute, IN 47803

dBug SIG
Dallas, TX
(214) 495-8828

**MASTERMENU™ ONLY $37**

**Instantly create/delete entire menu's**
**• Built-in menu generator • Passwords**
**• and more . . .**
**(207) 439-5074**
**ATI, Lyndon Way, Kittery, ME, 03904**

Picture City PC Programming Club
P.O. Box 20342
Rochester, NY 14602

Wiregrass Microcomputer Society
P.O. Box 1082
Enterprise, AL 36331

Hawaii PC User Group
P.O. Box 22967
Honolulu, HI 96822

Multi-Mate User Group
18314-48 Avenue West
Lynnwood, WA 98037

To receive an information sheet, call 213/538-7482

**Rates:** $1 per word. Minimum of 50 words. Submit copy and payment. We will reduce/ enlarge your art to fit available sizes.

**Payment:** Full payment must accompany each order. Mail copy and payment to: *TechNotes* Classifieds Attn: Brenda Johnson-Grau, 20101 Hamilton Avenue, Torrance, CA 90502-1319

**Agency Commission:** No agency commission on regular or display classifieds.

**Closing Date:** Ads must be received 15th of the month two months prior to cover date of magazine (that is, November 15th is the deadline for the January issue).

**Frequency discount:** 10% discount given for four consecutive insertions

**Acceptance:** Publisher reserves the right to accept or reject advertisements for Classifieds at their discretion.

**Tear Sheets:** A tear sheet showing classified ad as printed is sent without charge to each classified advertiser.

**Phone orders:** Not accepted.

**User Groups:** Ads are free. Name and address only. Limited to available space.

300 Personals

**STRESS ON THE JOB?**

Overworked? Burnout?
Depressed? Backache?

**Cheer up! We prescribe Genifer for dBASE programming without tears.**

**Genifer's the easy to use application generator that creates perfect applications in minutes.**

**Not only can it save you time, cranking out code. Genifer allows you to sit down with a user, listen to his or her requirements, and bring up a high-quality prototype in under an hour!**

**Call today for your free demo disk.**

**Bytel Corporation, 1029 Solano Ave. Berkeley, CA 94706. (415) 527-1157**
**1-800-631-2229 (CA 1-800-541-3366)**

**SUBSCRIBE TODAY TO**

TechNotes

**AND SAVE 30% OFF THE COVER PRICE !**

Get Ashton-Tate's hottest source of technical know-how! Each monthly issue of ***TechNotes*** provides you with the most current, accurate technical information available from the technicians at our Software Support Center on all dBASE, Framework, and RapidFile products. Return this card today! Or subscribe by phone!

Call our toll-free numbers to subscribers: **(800) 331-4164** (in California) or **(800) 828-2514** (residents of other states). In Canada, call **(619) 487-3212**. Cover price is $6.

☐ 1 YEAR for $49.97 Save $22.03 ! ☐ 2 YEARS for $89.97 Save $54.03 ! ☐ 3 YEARS for $124.97 Save $91.03 !

Add $15 for Canada and Mexico. Add $25 for overseas.

☐ Bill me ☐ Check enclosed (Make payable to Ashton-Tate in U.S. dollars drawn on a U.S. bank)

☐ Visa: ☐ MasterCard:

Expiration date: Signature:

☐ New subscription ☐ Renewal

Name

Address

City State/Province Zip Code

Country Phone Number

Which product do you use? ☐ dBASE II (D2) ☐ dBASE III (D3) ☐ dBASE III PLUS (DP) ☐ Framework (FW) ☐ RapidFile (RF) Please allow 30 to 60 days for delivery of your first issue. 1000

**READER SURVEY**

We would like to make sure that ***TechNotes*** continues to serve the needs of our readership. Please take the time to complete this survey.

Which Ashton-Tate product do you use?

☐ dBASE II ☐ dBASE III/dBASE III PLUS ☐ dBASE III PLUS Lan Pack
☐ Framework II ☐ RapidFile

How long have you been using these product(s)? _____ years _____ months

How long have you been a subscriber to ***TechNotes***? _____

Would you be interested in subscribing to ***TechNotes*** on disk? ☐ Yes ☐ No

Do you download the programs in ***TechNotes*** from the Ashton-Tate SIG on CompuServe or The Source? ☐ Yes ☐ No

Please rate these articles on a scale from 1 to 5 for accuracy, usefulness,and relevancy:

| | Accuracy | Usefulness | Relevancy |
|---|---|---|---|
| Functioning Function Keys | ☐ | ☐ | ☐ |
| Wrapped-Column Reports | ☐ | ☐ | ☐ |
| Taking the Humbug Out of PROTECT | ☐ | ☐ | ☐ |
| A Hypertext System for FWII | ☐ | ☐ | ☐ |
| Finding Matching Parentheses | ☐ | ☐ | ☐ |

**IF YOU'RE MOVING**

Don't forget to tell us so that we can be sure your subscription to **TechNotes** and/or the Ashton-Tate Quarterly is uninterrupted. To process a change of address, we must have your previous mailing address (including the zip code) and your new address.

My new address is:

Name/Department

Company Name/Add'l Destination Information

Mailing Address

City/State Zip Country

Please affix your current mailing label(s) here. (If the label is not available, be sure to include your old address.)

Affix your current Ashton-Tate Quarterly mailing label here

Affix your current TechNotes mailing label here

To change your address on the Ashton-Tate Product Mailing List, necessary to receive upgrade and other important information about your Ashton-Tate product(s), you must send a separate change of address to: Ashton-Tate Customer Service, 20101 Hamilton Ave., Torrance, CA 90502-1319.

# CLAS

All descriptions c
have been provid
These products a
tested by Ashton-
in this section sh
endorsement of

## SOFTWAI
## APPLICA

**SOFSOLUTION**
**ATTENTION: G**
**Clipper Compati**
automatically co
program into Q
able format. Just
the program doe
debugging time!
ver" order Geni
Clipper" order G
support. $59.00
**To order by ph**
**send a check to**
**San Antonio, TX**

## USER GR

dBASE SIG
St. Louis User's
for the IBM PC
P.O. Box 1345
St. Louis, MO 63

dBASE Users Gr
Picadilly Cafeter
Centerville Rd a
Dallas, TX 7302

Modesto PC Use
P.O. Box 5122
Modesto, CA 95

Space Coast PC
P.O. Box 396
Titusville, FL 32

**CONSU**
**Acc**

Are you familia
understand sm
are you interes
services under
gram on behalf
more than 10,0
You will be res
clients, recomm
counting softw
client services.
your current b
service you pro
compromising y
Recoup initial i
and start addin
ately. Call or w
pion Business S
Golden, Co 804
1-800-243-2626

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

## BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 66 TORRANCE,CA

POSTAGE WILL BE PAID BY ADDRESSEE

TechNotes

**Subscription Service**
P.O. Box 3729
Escondido, CA 92025-0929

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

## BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 66 TORRANCE,CA

POSTAGE WILL BE PAID BY ADDRESSEE

TechNotes

**Reader Survey**
Ashton-Tate Software Support Center
20101 Hamilton Avenue
Torrance, CA 90502-1319

Attention: Brenda Johnson-Grau

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

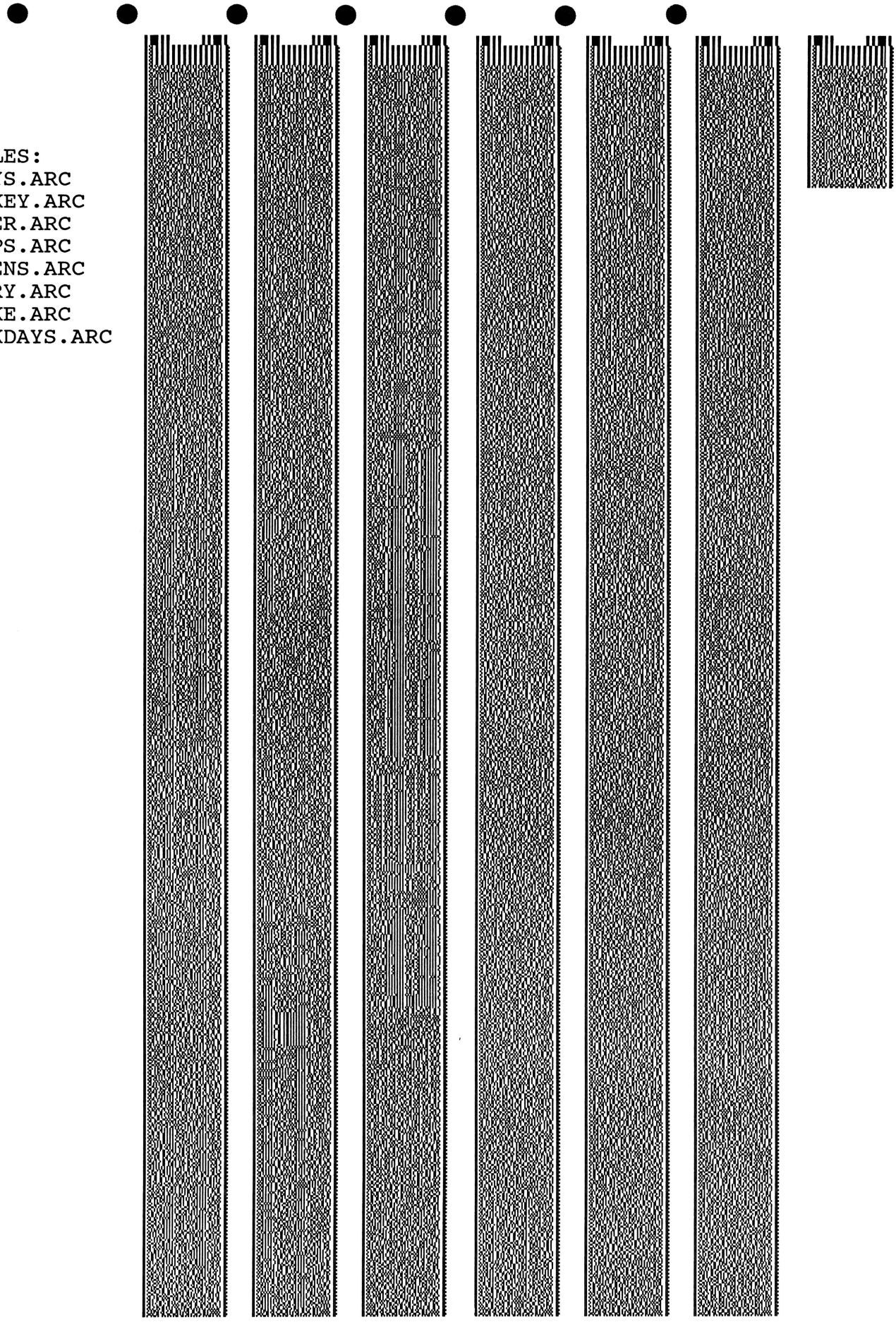
## BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 66 TORRANCE,CA

POSTAGE WILL BE PAID BY ADDRESSEE

**Ashton-Tate Periodicals**
PO Box 3729
Escondido, CA 92025-0929

FILES:
FKEYS.ARC
GETKEY.ARC
HYPER.ARC
ICAPS.ARC
PARENS.ARC
RETRY.ARC
SNAKE.ARC
WEEKDAYS.ARC



7 8 9 10 11 12

TechNotes
P.O. Box 3729
Escondido, CA 92025-0929

Address Correction Requested

BULK RATE
U.S. POSTAGE
PAID
PERMIT NO. 324
GARDENA, CA